National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

UNIVERSITY OF ALBERTA

Automatic Detection of Facial Features

By

Gloria L.P. Chow

©

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Masters of Science

Department of Computing Science

Edmonton, Alberta
Fall 1992

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-77119-4

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Gloria L.P. Chow

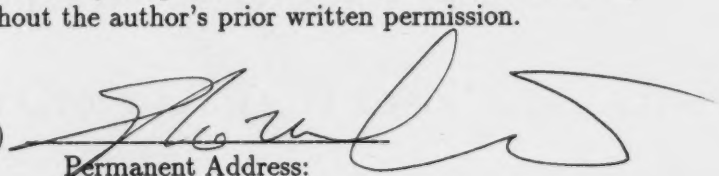
TITLE OF THESIS: Automatic Detection of Facial Features

DEGREE: Masters of Science

YEAR THIS DEGREE GRANTED: 1992

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

(Signed) 

Permanent Address:
119 Hill Drive,
Fort McMurray, Alberta,
Canada

Date: Oct 6, 1992.

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled *Automatic Detection of Facial Features* submitted by *Gloria L.P. Chow* in partial fulfillment of the requirements for the degree of Masters of Science.

Xiaobo Li

supervisor: X. Li

examiner: A. Basu

Roger W Toogood

external: R. Toogood (Mechanical Engineering)

Date: Oct 2, 1992

UNIVERSITY OF ALBERTA

Automatic Detection of Facial Features

By

Gloria L.P. Chow

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Masters of Science

Department of Computing Science

Edmonton, Alberta
Fall 1992

Abstract

The widespread use of photo-ID's for personal identification and security work indicates that there is a significant demand for an automated face recognition system. While there are many researchers working on the problem of face recognition, little work has been done so far on the automatic detection of facial features that is essential to the eventual recognition goal. This research is meant to be the initial phase of a completely automated face recognition system. Specifically, it will deal with the problem of automatic facial feature detection from a *loosely posed* head and shoulder picture of a clean shaven subject with no glasses. By loosely posed, we mean that the facial image is assumed to be a front-view id-type picture of the subject, but the face location, head size, lighting, and background may vary.

Research in psychology has shown that humans tend to recognize objects as a whole, rather than recognizing individual features. Our approach attempts to imitate this process by initially searching an image for shapes that roughly fit the template of a face. The facial features are then inspected to confirm or reject the template selected. This is different from the conventional sequential approach. The elimination of dependancy among feature modules yields a more robust and efficient system, since errors will not be accumulated and the system can be easily implemented in parallel.

The system can be roughly divided into two distinct levels of processing: speculation and confirmation. The speculation stage consists of a single context module that generates hypothesized face locations (ie. face contexts). The emphasis is on locating regions of interest roughly and quickly. The context module employs low level grey scale image processing techniques, such as morphological filtering and blob coloring to do rough analysis on a low resolution version of the input image. The detailed analysis is left for the confirmation stage modules. Currently, there are two confirmation stage modules: the eyes and the mouth. The purpose of both modules is to confirm as well as refine the location and shape of their respective features of interest. Therefore, in addition to low level processing as in edge detection, they must utilize higher level modelling approaches, namely the Hough transform and deformable template techniques.

Acknowledgements

I would like to thank my supervisor, Dr. Xiaobo Li, for all his guidance and infinite patience throughout this research. Special thanks must go to the coffee gang, Darin Ingimarson and Paul McWeeny, as well as Sergio Licardie, for the constructive discussions and supportive conversations I had with them. I am grateful to the members of my examining committee, Dr. Anup Basu, Dr. Stan Cabay, and Dr. Roger Toogood from the Mechanical Engineering Department, for their valuable comments, and also to my friend, Anne Gover, for the late nights she spent proofreading this thesis. Finally, and most of all, thanks to my fiancee, Don Murray, for giving me constructive suggestions when I was stuck and keeping me laugh when I was down.

Contents

1 Introduction 1

2 Automatic Facial Feature Extraction Literature Survey 5

2.1 Signature 5

2.2 Contour Following 7

2.3 Template Matching 8

2.4 Template and Spring Model 9

2.5 Hough Transform 10

2.6 Deformable Template 12

2.7 Difficulties in Facial Feature Extraction 16

3 System Overview 18

3.1 Testing and Programming Environments 20

3.2 Image Conventions 20

3.3 Preprocessing 21

3.4 Edge Detection 23

3.5 Morphological Filters 24

3.6 Thresholding 26

3.7 Geometric Primitive Generation 26

3.8 Function Optimization 28

3.9 Elastic Spring Functions 31

4 Context Module 34

4.1 Image Segmentation 34

4.1.1 Resolution Reduction 35
4.1.2 Valley Detection 35
4.1.3 Blob Coloring 36
4.2 Context Constraint Model 37
4.2.1 Pair Analysis 38
4.2.2 Context Completion 39
4.2.3 Context Evaluation 40
4.3 Experimental Results and Discussion 41

5 Eye Module **44**
5.1 Circle Hough Transform 45
5.1.1 Parameter Space Generation 46
5.1.2 Optimal Circle Selection 48
5.2 Deformable Template for Eye Boundary 49
5.3 Experimental Results and Discussion 52

6 Mouth Module **58**
6.1 Line Hough Transform 58
6.1.1 Parameter Space Generation 59
6.1.2 Horizontal Line Selection 60
6.2 Deformable Template for Mouth 61
6.3 Experimental Results and Discussion 62

7 Conclusion **69**

Bibliography **72**

A Modified Midpoint Circle Algorithm **75**

B Circle Arc Generation **78**

C Parabola Generation **80**

D Image Processing Library **84**

List of Figures

3.1 System Flowchart 19
3.2 Example Images 21
3.3 Image Conventions 22
3.4 Sobel Kernels 23
3.5 Laplacian Kernel 24
3.6 Star Mask 24
3.7 Spring Function Examples 33

4.1 Segmentation Step Flow Chart 35
4.2 5×5 Circle Mask 36
4.3 8-connected Blob Coloring Mask 36
4.4 Images Generated during Segmentation 37
4.5 Shape Resemblance Measure 38
4.6 Position Correspondence Measure 39
4.7 Context Component Search Regions 40
4.8 Module Region Extraction 41
4.9 Example Contexts 42
4.10 Failed Context 43

5.1 Bounding Box for the Eye Boundary Step 50
5.2 Eye Boundary Template 50
5.3 Eye Template Fit 54
5.4 Eye Template Energy Cost Distribution 56
5.5 Failed Eyes 57

6.1 Normal Representation of Line 59
6.2 Bounding Box for the Mouth Deformable Template Step 60
6.3 Mouth Template 61
6.4 Mouth Template Fit 64
6.5 Mouth Template Energy Cost Distribution 67

A.1 Eight Symmetrical Points on a Circle 76
A.2 Even and Odd Circles 76

C.1 Upright Parabola 81
C.2 Rotated Parabola 81

List of Tables

5.1 Eye Boundary Shape Function Coefficients 52
5.2 Eye Module Results 53

6.1 Mouth Shape Function Coefficients 63
6.2 Mouth Module Results 63

A.1 Generation of Symmetrical Points on a Circle 77

Chapter 1

Introduction

As society progresses and interactions among people increase, the potential of encountering an unfamiliar individual becomes a constant facet of modern day life. A passport grants us entry into another country. A bank card or account book allows one to withdraw money from an account. An entry access code lets an individual enter an otherwise prohibited, secured area. There is an obvious and widespread need of being able to identify and to be identified.

There is a multitude of media presently used for identification or security purposes. They range from possession of a simple artifact (eg. a key or a card), to special knowledge (eg. a password or memorized number sequence), or even to personal characteristics, either learned or innate (eg. signature, or voiceprint). None is more secure than a physiological trait, such as fingerprints, or facial images. Facial images are, however, by far the more popular and natural choice. It was no accident that the first postage stamp introduced in 1840's was a picture of Queen Victoria. As well, paper currency in many countries bear portraits of some important historical figures. Faces are used in these cases because they are permanent and unique to individuals. Their complexity also makes them extremely difficult to forge. One only needs to flip through his wallet to observe the widespread use of photo ID's. Identification by photo ID's is popular because everyone is considered an expert in recognizing faces and does it instantly.

However, psychology research has shown that recollection is a continuous process easily influenced by events after acquisition. In the case of eyewitness identi-

fication, the ability to recognize a certain individual from a group is significantly hampered by how many pictures are observed before the target [Wel88]. Furthermore, recognition, like other tasks requiring human attention, if performed repeatedly over time, is subject to degradation and occasional error. A security team of 2 guards for a company employing 200 employees may have to recognize over 400 faces on a typical work day. With today's mobile work force, change of staff is a frequent event. It would be impossible for the security guard to memorize every employee's face. Failure at times is to be expected.

Therefore, a completely automated face recognition system is obviously needed. The objective is not so much to completely alleviate the burden of recognition from humans, as we have yet to come up with an artificial intelligence approach that compares with the complexity, capability and speed of a human brain. Rather, the automated system should aid man by pre-screening all unlikely cases to ensure the integrity and enhance the speed of recognition.

Automatic recognition of faces by computer has been approached in two ways: constitutent-based and face-based. Much of the effort among the constitutent-based approaches has been concentrated on the examination of individual structural constituents of the face. These constituents includes objects such as eyes, mouth, nose, ears, etc. The feature vector used in comparison is extracted from measurements taken with and among these structural constituents. The primary advantage of this approach is its preservation of structural concept. It does not only help ease computational burden, but also carries an intuitive appeal because the measurements used are often physical quantities that one can in fact observe and appreciate. However, the relevance of these features in face recognition is yet to be proven objectively.

On the other hand, the face-based approach attempts to capture and define the face as a whole. A face is often treated as a 2-D pattern of intensity variation. The objective is then to discover its underlying statistical regularities. Much of the work in the area has been accomplished through the application of neural networks [KLO81] [Sto86]. There are also others which utilize standard statistical analysis techniques [Bar89] [TP89] [SK87]. The primary appeal of this approach is that the developer is spared the task of having to explicitly specify the *features*

to be used in the recognition process. The description will be automatically and objectively derived from the initial training set. Yet, because the facial image is perceived merely as a set of 2-D data, often the structural concept is lost and can no longer be taken advantage of.

However, before a face can be recognized, it must first be extracted from an image. This detection task is necessary regardless of whether one intends to use the face-based or constituent-based recognition approach. For the constituent-based approach, since the feature vectors will be based on actual measurements taken from and between facial constituents, it should be obvious that a reliable means of detection is fundamental to the success of the overall system. For the face-based approach, though the knowledge of the locations of facial constituents is not necessary for the actual recognition process, it is often desirable so that input can be normalized to a standard size, location, light exposure and orientation. Of course, this could be done manually during the image capture phase. It would take meticulous adjustment that is difficult to guarantee for more than a handful of images. The alternative is to normalize based on some important landmark from the image itself. A typical choice is a measurement taken from some facial features, such as distance between nose and mouth, intensity from cheek patch, angular distance between eyes, etc. This will necessitate the detection of some facial components, just as that required by the constituent-based approach. While there are many researchers working on the problem of face recognition, little work has been done so far on the automatic detection of facial features that is essential to the eventual recognition goal.

The reader should note that there has been much research done using profile rather than frontal images [Har76]. While profile images convey better information regarding the degree of protrusion of various facial components, frontal images provide a better view of the facial structure as a whole and are better related to by a human observer. Furthermore, it is by far the most popular view used in personal identification pictures.

This research is meant to be the initial phase of a completely automated face recognition system. Specifically, it will deal with the problem of automatic facial feature detection from a *loosely posed* head and shoulder picture. By loosely posed,

we mean that the facial image is assumed to be a front-view id-type picture of the subject, but the face location, head size, lighting, and background may vary. In specific, we will attempt to extract the locations and shapes of eyes and mouth for each input image using low level grey scale image processing techniques such as morphological filtering, and edge detection, as well as higher level modelling paradigms, namely Hough transform and deformable template. Our proposed system will allow features to be extracted in parallel rather than sequentially so as to minimize the potential of error accumulation, but yet maximize the system efficiency.

In order to simplify the problem, the subject is assumed to be clean shaven and without glasses, though images to the contrary are occasionally used to test the robustness of the system. In addition, while facial expression is an important element that will influence the shapes of many facial components, it is such an intangible concept that little research has been pursued in the field as yet. Since the primary motive of our research is for applications in personal identification where relatively little facial expression is expected to be present, the analysis of facial expression is deemed beyond the scope of this study.

The remainder of this thesis will be devoted to document the justification, methodology, and results of our proposed approach. Chapter 2 is a review of the techniques used by past researchers in the detection of facial features from frontal images. Chapter 3 contains an overview of our proposed strategy, as well as discussion of some general techniques used throughout the system. Chapters 4 to 6 detail the development and experimental results of the various modules in the system. Finally, chapter 7 summarizes the research presented and future research directions.

Chapter 2

Automatic Facial Feature Extraction Literature Survey

In most literature in automatic facial recognition, relatively little effort has been spent on the facial feature extraction problem. For many [GHL71] [CJCM86], it was because extraction was done manually, on the assumption that the problem would be resolved somehow. For others [WLT89] [Con86], their aim was to *recognize*, and therefore even though attempts of automatic extraction were made, little was reported on the success of it alone. In the following sections, we shall review some of these methodologies and evaluate them qualitatively. The approaches used reflect closely the development of object recognition techniques in computer vision over the past 20 years. Facial feature extraction is after all just an instance of the general object recognition problem. Therefore, we have organized the review into categories by technique, in addition to an overall review summary as follows.

2.1 Signature

The *signature-based* technique aims at encoding shape by projection. Projection here refers to the act of summing along a particular direction. The quantity to be projected could be the actual intensity, or edge strength, or some other meaningful image parameters. The maxima and minima in the projection can be used as an

indication of the presence or absence of certain landmarks in the image.

In the work by Sakai *et al.* [SNK72], the basic assumption was that each feature exhibited a unique, easily identifiable distribution profile of edge pixels. By recognizing certain patterns of distribution, Sakai *et al.* were able to detect facial features in sequential order. The resulting technique was called *integral projection.* Under this technique, edge pixels within a slit of proper width and length were summed column-wise to obtain a distribution profile. This slit was slid around the image until the desired pattern was detected. For example, the top of the head was simply described as the first significant output within a horizontal slit sliding down from the top. The sides of the face were identified similarly through another horizontal slit. However, this time it would exhibit a sequence of patterns rather than a single one. The sequence would correspond to when the slit crossed over the eyes, the eyes and the nose simultaneously, and finally just the nose. Throughout this sequence, the sides of the face could be identified as the two edge maxima bounding the eyes and nose. Once the top and sides had been identified, searching for internal features could be limited to a smaller area. The same technique was used to locate nose, mouth, chin contour, and eyes by varying the shape, size, and orientation of the slit.

Based on a similar idea, Lambert developed the *signature* search technique to locate facial features [Lam87]. A signature here was defined to be the vertical intensity profile, obtained by summing the intensity in each column. The first to be located were the eyes whose signature consisted of three brightness maxima (bordering the eyes) and two brightness minima (at the eyes). Once the eyes were located, searches for the mouth and nose could proceed in limited area. This approach was very much similar to the work of Sakai *et al.* [SNK72], the only difference was that grey level intensity rather than edge pixels was used.

Cannon's approach [CJCM86] was based on a combined principal of the previous two [SNK72] [Lam87]. The top of the head was first located by searching for a significant intensity gradient from the top downward. The sides of the face were found by searching inward at a predicted range of height, based on the top of the head, for a significant positive gradient. Subsequently, the two cheek patches were located by searching for intensity maxima within the predicted locations based on

the previous two steps. The eyes were found similarly by searching for intensity minima above the cheeks.

The primary advantage of this approach is its simplicity, not accuracy. It is easy to implement and quick to execute, making it very attractive as a preliminary filtering step to reduce subsequent search effort. Nevertheless, projection is not an information-preserving transformation. There may exist many distinct pre-transformed images for any one unique pattern of projection. Therefore, the presence of a certain projection pattern by itself is not an absolute indication of an object instance. With a uniform background and plain clothing, there are only a few objects in the image, and the shape and image characteristics of these few objects will be known *a priori*. Success can be moderately assured. Yet, with a cluttered background and arbitrary clothing such as that found in most photographic sessions, it will be difficult to guarantee that a certain projection pattern is unique for one and only one object or set of objects. Therefore, the signature-based technique, though useful as a preliminary step, must be augmented with other techniques or multiple-projections to reliably extract objects in a more general case.

2.2 Contour Following

In attempting to overcome the uniform background restriction, Craw and Ellis used a multi-resolution template-guided contour following scheme to separate the foreground (ie. the head outline) from the potentially cluttered background [CEL87]. At its lowest resolution, a predefined template of a head and shoulder outline was used as a guide for the contour follower to extract a gentle curvature outline. The contour found at one level was then used as the template in the next higher resolution level. The shape was therefore gradually refined through successively higher levels of resolution, until the maximum resolution was achieved.

Once the head had been identified, locations of the other components would be roughly known. Thus, they could be modelled using relatively simple *signatures*. The lips and eyebrows, for example, were approximated as horizontal lines. Here, the lips were searched for first along the vertical line down the center of the image.

Once the lips were located, the eyebrows should intersect the vertical line passing through either corner of the mouth. The eyes, on the other hand, were simply a pair of local minima under the eyebrows.

Contour following, a pixel-by-pixel operation, is extremely susceptible to noises and gaps in the image. The addition of a template model was designed to add global constraint to an otherwise local operation. Unfortunately, the assumption of slow curvature change was often found to be violated in a real image at small scale. Since it was the initial step, there was little other reliable information available to correct or sidestep any inconsistency encountered during the process. Because of the sequential nature of processing, any errors incurred early on would be accumulated and possibly amplified further down. Therefore, the overall rate of extraction was rather poor. Only 12 out of 20 of the head outlines were successfully extracted. For those which failed, many were unable to continue because of a lack of reliable reference.

In summary, because of its susceptibility to noise, the contour following technique is better reserved as a detail shape extraction operation at a latter stage after a reliable reference has been established through some other means.

2.3 Template Matching

The basic idea of template matching is to find an area of high correlation between a template (ie. a fixed 2-D pixel pattern) and a sub-region of the same scale in the image. The correlation is done on a point-by-point level and is measured by the number of matched pixels.

Baron used this technique to locate the eyes [Bar89]. A number of eye templates were abstracted from real images and used in the matching process. The eye was located by finding an area of sufficiently high level of correlation with any one of these templates. The objective was to use the distance between eyes to normalize the input image for further statistical pattern-recognition processing. Though the rate of recognition was reported to be very impressive, there was no comment on how accurate and robust the eye extraction module was. He did report that large changes in image scale could not be accommodated adequately

despite the scale standardization procedure.

This result is to be expected since conventional template matching requires the shape and proportion of the object of interest to be precisely known. The rate of correlation is directly proportional to how closely the object in the input image resembles the template, both in scale and orientation. Therefore, even if the object and the template are identical in shape, if one is larger than or rotated slightly from the other, the correlation will be poor. Furthermore, few objects in nature have a rigid shape. Instances of the same object class may possess similar *overall* shape, but the aspect ratios among components could vary considerably. It is difficult to determine how many fixed templates are needed in order to fully represent all shapes within the class. Consequently, given an object instance which is inadequately represented in terms of scale, orientation, and slight shape variation, it will be difficult to place the existing template precisely, as is the case found here by Baron [Bar89].

The alternative is to increase the number of eye templates used, hoping to cover most situations. Correlation, by itself, is $O(n^2m^2)$ process where n^2 and m^2 are the sizes of the input image and template respectively. If the number of eye templates is large, the computational expense will be considerable. In short, template matching is a useful extraction technique if the size, shape and orientation of the object can be limited, but it is largely impractical for general application.

2.4 Template and Spring Model

A fixed template can be made more flexible by adding the spring concept. The underlying idea is to model an object as a collection of distinct components, each of which can be viewed as a rigid template. The relations among these components are expressed as spring tension functions. An ideal instance of the object will have a total tension of zero. As the relations among components deviate from the ideal, the total tension increases quadratically. It thereby provides a means of measuring the quality of fit of the template to the image.

Govindaraju *et al.* used this template and spring model to extract head out-

lines from newspaper photographs [GSSS89]. The head outline was modelled as three separate curve segments: top, left, and right. The extraction of these curves was done by edge detection, skeletonization and relaxation linking. Then the curves were grouped by spatial constraints to form potential head outlines. Each potential head outline was evaluated by a combination of object component cost, and relation spring cost. There were three terms in the overall cost function:

1. $C(temp)$ measured the dissimilarity between the template and the combined outline from the three curve segments.

2. $C(miss)$ was the penalty of any missing curves.

3. $C(spring)$ measured the connectivity between any two adjacent curves.

The system was tested only on 10 images, many with more than one instance of head outline. In all cases, it succeeded in finding all faces present which indicates the system is relatively robust. However, false alarms were often generated. As with the signature-based technique, this approach has the problem of having a many-to-one mapping between the actual object and model. Here, basically any elliptical object, regardless of internal components, will be identified as a potential head. Because of better shape representation, this is more robust than the signature-based approach. This improved performance does have to come with considerable computational effort since the extraction phase is much more complicated. Nevertheless, as with the signature-based approach, this technique must be augmented with other detailed component analysis in order to be used with high confidence.

2.5 Hough Transform

The Hough transform is an edge-based segmentation technique for detecting edges whose shapes can be described by parametric curves (eg. straight line and conic). The basic idea is selection by counts. If there exists an edge of the desired analytically defined shape, the points along this edge will all have the same set of parameters. Therefore, if we maintain a count of corresponding edge points

for each possible set of parameters, the truly existed curves can be extracted by identifying those with a high accumulation count. This array of accumulators is collectively known as the parameter space. The number of dimensions within the parameter space is simply the number of parameters needed to express the curve analytically. Given the inverse analytic equation of a parametric curve, F, in order to transform edge points from the spatial domain to that in the parameter domain, we shall assume each significant edge pixel forms part of the boundary of the desired curve. Knowing the coordinates of this point (x, y) and the inverse function F, we can solve the parametric equation backward to obtain the corresponding parameters and increment its accumulated count in the parameter space. Since there are usually far fewer image points than the number of cells in the parametric space, the number of function evaluations is drastically reduced. This is essentially an efficient implementation of the generalized template matching and is very robust even in the presence of noise.

Nixon [Nix85] examined the effectiveness of Hough transform for extracting eye spacing measurements in three different models:

1. a circle approximating the shape of the iris.
2. an exponential function tailored ellipse approximating the boundary of the eye.
3. the same tailored ellipse approximating the combined area of the eye and the region below the eyebrows.

In all three cases, matching was done over only a very small area around the eye to limit the search and avoid accidental matches with objects other than the eyes. However, he did not address how this rough location could be derived. The detection performance reported was extremely promising. He found that measurement by detection of the position of the iris was the most accurate, with less than half a pixel difference between the detected distance and the manually derived distance. Detection of the eye boundary, on the other hand, was the most sensitive to noise, but did offer a similar degree of accuracy under good conditions.

Govindaraju *et al.* also used a modified Hough transform approach in his second attempt at extracting head outlines from newspaper photographs [GSS90].

They divided the head outline into four pieces composing of 2 arcs for the top and bottom, and 2 lines for the sides of the face. The extraction of these was done through first level Hough transform units designed for detecting lines and elliptical arcs. Then, a second level transform based on the Hough transform idea was performed on the extracted lines and arcs. Here, an ellipse was defined for every potential top-of-the-head elliptical arc, and a collection of such ellipses would form the parameter space. Every line or arc contributed only to the ellipse which it intersected the most. Therefore, potential head outlines could be extracted by tallying the votes for each ellipse and selecting only those with high counts. It was also suggested a face-verification stage would be needed in order to confidently confirm that the selected candidates were indeed heads. Unfortunately, this verification stage was still under development and there were no performance statistics quoted for the head candidate generation stage.

In general the major drawback of using Hough transform is, of course, its computational demand. The computational requirement typically grows exponentially with the number of parameters in the analytic equation. Though Nixon had mentioned using the 8-symmetry property in generating circles, there was no remark of similar strategy for easing the computational burden in generating the tailored ellipse. The same was true for Govindaraju's elliptical arc extraction unit. Therefore, it is difficult to conclude whether these methods are practical means for extracting facial features.

2.6 Deformable Template

In contrast to previous techniques, the deformable template approach uses an *active* model in the sense that its shape is not fixed and it evolves over time to best explain the existing image conditions. A deformable template is basically an arrangement of simple parameterized geometric primitives such as straight lines, circles, and parabolas, to delineate the general shape. Altering the template parameters corresponds to changing the position, orientation, size, and feature proportion of the object. Therefore, a family of similarly shaped and sized templates can be compactly represented by one deformable template. The task of

determining the ideal set of parameters to be used is accomplished through numerical optimization. The function to be optimized is an *energy* function designed to measure how well a parameterized template corresponds to the input image, and how closely it conforms to an average preferred shape. Image correspondence is often measured in terms of typical image forces such as edge, valley, peak and intensity. Because different image forces influence the energy function in different manners, optimization is often done over a number of epoches (ie. stages). For example, valley and peak are usually emphasized in early epoches to pull the template in from a distance away providing there is no major distraction on the way. On the other hand, edge and intensity forces are better at fine tuning the template in the latter epoches.

The primary attractiveness of this approach is its strong representational power. An object can be thought of as a number of well-structured geometric primitives, controlled by only a handful of parameters rather than a series of points which can be very loosely connected with little coherence from endpoint to endpoint. Of course, one may argue that a pointwise model is a more precise representation on a local level. Yet, with the parameterized model, global non-accidental properties such as symmetry, parallelism, and junctions are much more readily available. Psychology research has shown that they are more reliable cues for visual input interpretation [Bie87]. The parameters have global rather than local effect, and hence the model will be less susceptible to incidental noise and gaps, which are prevalent in real images. Furthermore, since there are now only a handful of parameters to manipulate as opposed to a series of individual point coordinates, the combinatorial search space is drastically reduced.

These advantages, of course, are equally true for the Hough transform technique. Nevertheless, unlike the Hough technique, deformable template technique does not require the entire parameter space to be generated. This translates into tremendous saving in terms of storage and computations. It is feasible to represent a much more complex structure than that allowed in practicality by Hough transform. The actual efficiency will depend upon how the parameterized template is evaluated, and how its corresponding energy function is optimized.

On the other hand, the Hough transform, with its high cost, will always gen-

erate the optimal solution. As for the deformable template approach, because a global optimization technique is yet to be discovered, the solution found by using existing optimization techniques may be *satisfactory*, but not necessarily the best. The choice of the modelling technique used therefore requires a tradeoff between accuracy and practicality.

This technique was first purposed by Yuille *et al.* [YCH88] for extracting eyes and mouths. The eye was modelled with a circle (iris), a pair of bounding parabolas (eye contour), and two points (centers of whites of the eyes). The corresponding energy function was composed of five terms: edge, valley, peak, intensity, shape. The optimization was done over six epoches varying different image forces in this order: the overall valley; the intensity and then the edge over the circle; the peak and then the intensity over the whites of the eye; and finally the edge force over the bounding parabola.

Two different templates were used for the mouth, one for the mouth-closed case, and the other for the mouth-open case. The mouth-closed template was defined with four parabolas: two on the bottom for the lower lip and center separation line, and two symmetrically placed on the top to mark notched upper lip. The mouth-open template was similar but with one additional parabola in the middle so that the upper and lower lips no longer had to share the same center separation line. The energy function had only three terms: edge, valley, and shape. The optimization was done over two epochs: first over the valley for pulling the template in; then over the edge for fine tuning.

Yuille's paper [YCH88] was primarily an introduction and feasibility study. There were no precise statistics on the extraction performance. However, it was reported that some tuning was required to set up the energy term coefficients, and providing the eye template was started below the eye, it would lock on to the eye. In both, run times were reported to be between five to ten minutes. This timing result was measured after calculation of all related image forces (eg. edge, valley, peak, etc.) were completed.

Shackleton *et al.* used the same eye template to extract eyes from facial images [SW91]. The energy function, however, was slightly different. An enhanced whites term was added to replace the intensity over original whites term. They also found

that the shape energy term and the minimization epoches had to be modified. In terms of extraction performance, Shackleton *et al.* reported that 53 out of 60 eye images were fitted to some degree. Forty-one of these were very good to moderate fit. Once the parameters to the eye template were found, they were used to geometrically normalize the eye image. A set of eye *eigenvectors* was extracted from a training set of transformed images via Principal Component Analysis. All eye images were encoded in terms of these eye eigenvectors for recognition purpose. Out of a set of 24 test images, 16 were correctly identified and of the remaining 8 images, 5 included the correct identification within the best 5 matches. The result was expected to improve if other components of the face were recognized as well.

Craw *et al.* also used the deformable template technique in his mug shot retrieval system [CTB92]. Their eye module employed an eye template similar to that used by Yuille *et al.* and Shackleton *et al.* , but with a slightly varied energy function and minimization technique. In addition, the head outline was extracted using similar deformable template technique. A polygonal template was used to approximate the head outline based on edge information only. Optimization was done over two epoches. In the first epoch, the template was allowed to deform only in terms of position and aspect ratio. This rapidly drew the template to the rough head location. This location was further refined in the second epoch where a higher degree of variation was allowed. Here, individual vectors that made up the template were allowed to deform in terms of scale and orientation. Deformations between adjacent vectors were coordinated to ensure that a closed and head-like polygon was achieved. The process was rather complex and time consuming, but it was capable of producing reliable results given an unrestricted search area.

Both of these are just parts of a more complete system. There were two unique concepts in this system. First, Craw *et al.* used a hypothesis and verification scheme akin to that proposed by Govindaraju [GSS90]. Modules could be classified into two groups where one's aim was to hypothesize some point of reference, and the other was meant to verify the validity of the hypothesis. For example, the head outline module was used to establish the reference, and the

eye module was used to verify it. Second, they employed a redundancy concept throughout the system. Often, there were more than one module accomplishing the same task so that if one failed, the other might step in. For instance, a default template extracted statistically from a training set could be used to establish an initial reference in place of the complex head outline module. As well, the head outline can be roughly extracted by a module which returns all viable combinations of blobs configured as two eyes and mouth. Unfortunately, in their paper, Craw *et al.* only discussed the eye and head outline modules in detail, and many of the other modules were mentioned only in brief.

2.7 Difficulties in Facial Feature Extraction

As we can see the problem of facial feature extraction is far from being satisfactorily resolved. One may ask what makes it so difficult. Here, we shall first examine the inherent problems of computer vision and then see how they apply to our problem in particular.

The sources of difficulties of many vision problems are perhaps best summarized by Rosenfeld *et al.* who spoke of the three natures of vision problems: *ill-posedness*, *ill-definedness*, and *intractability* [RC92]. The goal of vision is to hopefully correctly interpret the available visual input. It is essentially an inverse mapping problem, which if not properly constrained, may have multiple solutions. Ill-posedness is referring to this underconstrained situation. In order to force it into a well-posed problem, external constraints can be added. Examples of this are *a priori* knowledge or assumptions about geometric structure and specific image characteristics of the objects and its surrounding. When these constraints do not reflect reality, we have an ill-defined problem. In general, any problem which is not adequately expressed in precise terms is considered ill-defined. Even if it is well-defined, it is often impossible to solve it analytically. The alternative then is to solve it through empirical means (ie. enumerating all possible solutions and evaluating it one by one to extract the best). Depending on the size of the solution space, finding a solution could be an extremely expensive and intractable process.

The varying degree of success and failure of the techniques reviewed in the last section is a reflection of how well they handle these three problems. Foremost, a face is a very complex object. It is difficult to describe in words, let alone define it in precise mathematical terms. As we have shown, few techniques reviewed have addressed the face extraction problem as a whole. Rather many treat it as a number of sub-problems of identifying individual components. Others simply concentrate on the extraction of a single component. We should examine how these components are represented in order to determine whether the ill-posedness clause applies. In general, we have observed that the simpler the model, the more likely that there are alternative interpretations. This explains why the signature-based approach is the most ill-posed, and it improves as one moves from fixed template modelling to template and spring model, Hough transform, and finally deformable template. Yet, an unconstrained search of an instance of a complex model is computationally expensive and potentially intractable. Therefore, the extraction module must be localized. For some researchers, this localization problem is left unaddressed [Bar89] [Nix85] [SW91] [YCH88]. For others, they choose to use the sequential approach, so that one feature serves to limit the search for the other features [SNK72] [Lam87] [CJCM86] [CEL87]. Unfortunately, this places unrealistic demand on the robustness of the initial component extraction. In order to make it reliable, one must use a more complex model. Yet, that makes the problem intractable and we have come around a full circle without a satisfactory solution. Of course, some [GSS90] [CTB92] had gone ahead and used such complex model to establish their initial reference. We are not sure just how much was paid computationally to obtain the necessary reliability. Nonetheless, they both realized that there should be a verification stage by other means to ensure overall system integrity.

The alternative is to start with a simple model, and resolve the ambiguity through other more complex modules later. The advantage is that by then the search area can be limited for the more complex modules. Craw *et al.* mentioned the use of such a simple approach in [CTB92]. It is however unclear how this was implemented and whether it was successful.

Chapter 3

System Overview

Research in psychology has shown that humans tend to recognize objects as a whole, rather than recognizing individual features. Our approach attempts to imitate this process by initially searching an image for shapes that roughly fit the template of a face. The facial features are then inspected to confirm or reject the template selected. This is similar to the approach used by Craw *et al.* [CTB92]. However, we will attempt to keep the initial hypothesis generation module simple, and only use complex modules later for verification purpose. Hence the system can conceptually be divided into two separate stages: speculation and confirmation.

In the speculation stage, we establish a *facial context.* A *facial context* is a collection of distinct regions whose spatial arrangement resembles that of the eyes, eyebrows and mouth. In essence, we want a rough guess of where the face is located in the image.

Then in the confirmation stage, we attempt to verify whether the proposed context is accurate. This is accomplished by passing different regions to their corresponding extraction modules to be analyzed in detail. Currently there are only two feature modules: the eyes and mouth. The general strategy in both is identical. First, we model the desired feature using simple geometric elements such as straight line, circle, and parabola. Then, we use Hough transform and deformable template techniques to extract features within the given region that best match the model.

Figure 3.1 is a system flowchart illustrating this idea. The solid boxes are the

modules which are actually implemented in this system, while the dashed boxes are potential modules which could be added. The arrows on the right side indicate the data flow between the two levels. The speculation module passes processing region information to each confirmation module. In return, each confirmation module will provide a verification, as well as shape and location information back to the speculation module. The important design objective here is to eliminate

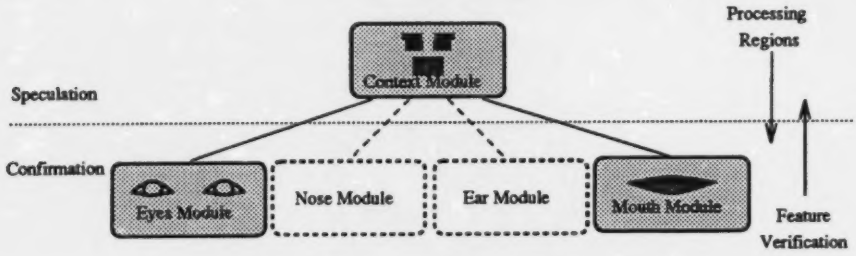


Figure 3.1: System Flowchart

dependency among confirmation modules, so that if one fails, the rest can still continue. This is in sharp contrast to most previous work in facial feature location. They primarily employ a sequential approach, so that the successful location of one feature will serve to limit the search for the subsequent features. While it is computationally attractive, the inter-dependency among modules leaves the system far too unreliable. Another advantage of our design over a sequential approach is the potential gain in speed. While speed is an issue seldom discussed in most face recognition research, it is an important consideration in any practical system. In our design, since the confirmation modules are completely independent from one another, it will be relatively easy to port such a system into a large grain parallel processing paradigm to achieve performance speedup. This is a goal that will be extremely difficult to achieve with the conventional sequential approach. In the remainder of the current chapter, we shall discuss the conventions and general techniques that are used throughout the system, so as to prepare the reader for detailed description of individual modules in the later chapters.

3.1 Testing and Programming Environments

All the modules described in this thesis were implemented in "C" language. Most routines were designed as general purpose procedural elements, collectively forming an image processing "C" library. Appendix D provides a summary of this library. All timing results were reported from tests run on a Sun 25MHz CMOS Sparc IPC with a benchmark performance of 13.4 SPECmarks89. [1] The test images consisted of frontal head and shoulder shots of students in the Computing Science Department at the University of Alberta. These images were captured using a Panasonic TV camera, and digitized with a VideoPix frame grabber board. All images were grey scale images of size 256 × 256, and stored in Sun raster format. Two sets of images were taken over a period of one year. The majority was from the first set taken early on in the year, with a dark uniformly colored cloth as backdrop. Because of the low quality sensor and inexperienced operator (namely the author), many of these images were characterized with uneven lighting and poor contrast. A smaller set was taken later on. They were generally of better picture quality, and this time a non-uniform background was introduced to add realism. Figures 3.2 illustrates an example for each set.

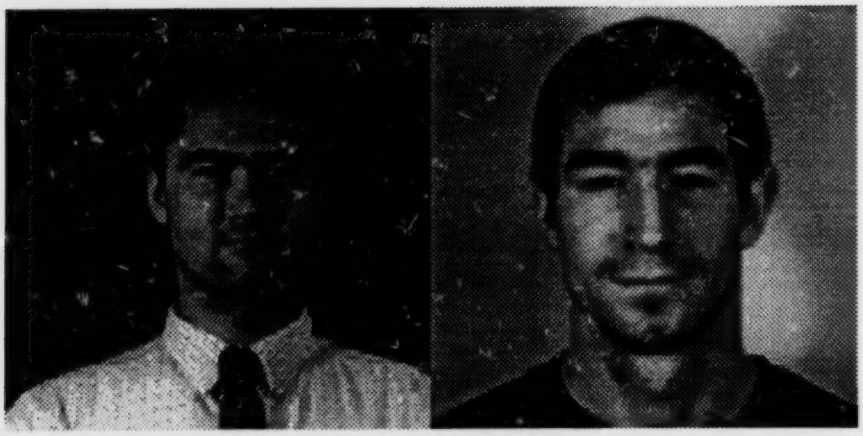
3.2 Image Conventions

In general, an image is considered to be a two dimensional sequence of real numbers, $I(x, y)$. The indices x and y are limited to some fixed intervals by their respective dimensions, X and Y as follows:

$$0 \leq x < X$$
$$0 \leq y < Y$$

The Cartesian coordinates will assume the usual display device conventions, namely starting from the upper left corner, and proceeding to the right and down. All angles are expressed in terms of the usual polar coordinates conventions. They are measured in reference to a directed half-line pointing to the right, and are

[1] This is in reference to a DEC VAX 11/780 which has a performance of 1 SPECmarks89.



(a) Example from Initial Set (b) Example from Latter Set

Figure 3.2: Example Images

positive rotated counter-clockwise, and negative for the opposite direction. In addition to these conventions, all input and output grey scale images are assumed to have integer grey level values between 0 and 255, with 0 representing black and 255 representing white. These conventions are summarized in Figure 3.3.

3.3 Preprocessing

The objective of preprocessing is to enhance or filter an image so that better results can be obtained in subsequent steps. To enhance edges, one may choose to amplify the high frequency components. At the same time, noise is often assumed to be from the high frequency spectrum. In order to remove it, one may have to depress the high frequency response. By removing high frequency noise, one may inadvertently delete a certain amount of detailed information (ie. the high frequency component) from the original image. These are conflicting objectives and are difficult to resolve satisfactorily.

As well, our proposed system will involve a number of different techniques operating on different levels of image detail. Some will work on a sampled version of the original, while others will operate on a sub-region of the original. Trying to

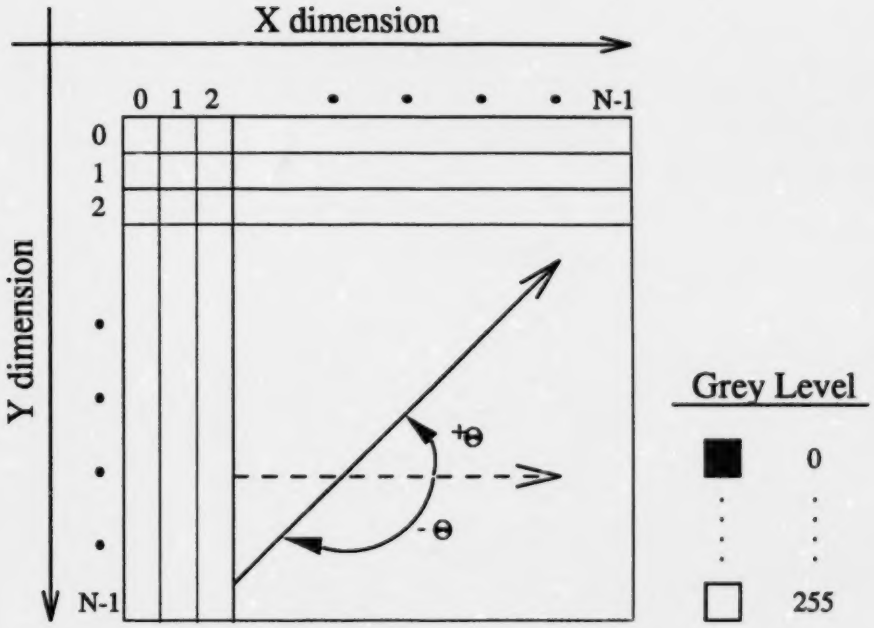


Figure 3.3: Image Conventions

design a preprocessing operation that is beneficial at all levels is not only difficult but is also computationally wasteful. Therefore, no overall preprocessing is done, and each individual module shall assume the responsibility of preparing the image for its own purpose.

3.4 Edge Detection

Edge detection is done by convolving the image with two Sobel kernels in the spatial domain to detect grey-level discontinuities. The two Sobel kernels are 3×3 masks as shown in Figure 3.4

-1	0	1
-2	0	2
-1	0	1

(a) X kernel

1	2	1
0	0	0
-1	-2	-1

(b) Y kernel

Figure 3.4: Sobel Kernels

Convolution with the Sobel kernels shall yield the x and y components of the gradient vector. The overall magnitude of the gradient vector can then be computed as:

$$G[f(x,y)] = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{3.1}$$

where G is the magnitude of the gradient vector,
G_x is the result of convolution with the Sobel x-kernel,
G_y is the result of convolution with the Sobel y-kernel.

As well, the direction of the gradient vector, α, can be obtained by:

$$\alpha(x,y) = tan^{-1}(\frac{G_y}{G_x}) \tag{3.2}$$

where G_x and G_y are the same as before.

Equations 3.1 and 3.2 will compute a gradient vector at every pixel location, regardless of whether an edge in fact exists. Though the magnitude will serve to

confirm edge validity in most cases, the precise edge location can be estimated by detecting zero crossings of the second derivatives of the 2-D intensity function. The second derivative can be approximated by convolving the image with the 3×3 Laplacian kernel as shown in Figure 3.5. Zero crossings can then be identified by

-2	1	-2
1	4	1
-2	1	-2

Figure 3.5: Laplacian Kernel

locating all positive to negative transitions.[2]

3.5 Morphological Filters

Morphological filtering is a set theoretic approach to image processing. It transforms the pixel value of an image through set operations on points within its neighborhood. The extent of this neighborhood is defined by a *structural element*. A *structural element* is a small mask of simple shape, such as circle, square, rhombus etc.. In our case, it is implemented as a $M \times N$ rectangular mask of binary values. The binary values are used to encode shape information: 1 indicating the interior, and 0 the exterior. Figure 3.6 is a star mask that may be employed for morphological filtering purposes.

0	1	0
1	1	1
0	1	0

Figure 3.6: Star Mask

[2]Theoretically, both positive-to-negative and negative-to-positive transitions may denote valid zero crossings. However, in order to avoid double lined edges, only one is used here.

The concept of morphological filtering is similar to spatial convolution. We define an $M \times N$ mask and overlay it on the image so that the center of the mask moves from pixel to pixel. At each location, we replace its value with the result of a function, f, evaluated over all the points under the mask. In the case of spatial convolution, f is a weighted sum function, whereas in morphological filtering, this function is limited to some logical set operations.

In general, morphological filtering can be operated on either binary or grey scale images. A general treatment of both can be found in [Ser82], [Vog89], and [Mar87]. For our purposes, we shall limit the discussion to that on grey scale images only. There are four fundamental operations in morphological filtering: dilation, erosion, opening, and closing. If we have a grey scale image X, and a $M \times N$ structural element S, the morphological erosion $\ominus$ and dilation $\oplus$ operations can be expressed as:

$$X \oplus S(x,y) = \begin{cases} \max_{m=0}^{M-1} \max_{n=0}^{N-1} X(x-m+\frac{M}{2}, y-n+\frac{N}{2}) & \text{if } S(m,n)=1 \\ \text{ignore} & \text{if } S(m,n)=0 \end{cases}$$

$$X \ominus S(x,y) = \begin{cases} \min_{m=0}^{M-1} \min_{n=0}^{N-1} X(x-m+\frac{M}{2}, y-n+\frac{N}{2}) & \text{if } S(m,n)=1 \\ \text{ignore} & \text{if } S(m,n)=0 \end{cases}$$

where max and min are the local maximum and minimum operators respectively. In other words, the erosion of a pixel is the minimum pixel value of all points under the structural element centered at that pixel; and vise versa for the dilation operation.

The opening $\circ$ and closing $\bullet$ operations are defined as combinations of erosion and dilation as follows:

$$\begin{aligned} \textit{Opening}: \quad & X \circ S = (X \ominus S) \oplus S \\ \textit{Closing}: \quad & X \bullet S = (X \oplus S) \ominus S \end{aligned}$$

Opening reduces the intensity peaks by flattening it to the limit of the structural element. Closing raises the intensity valleys. If one then compares these results with the original, he will be able to identify locations of intensity peaks and valleys. This is the so-called residue operation and can be expressed formally as

follows:

$$\begin{aligned} \textit{OpeningResidue}: \quad & X \overset{\circ}{-} S = X - (X \circ S) \\ \textit{ClosingResidue}: \quad & X \overset{\bullet}{-} S = (X \bullet S) - X \end{aligned}$$

where $-$ refers to simple image subtraction. The primary advantages of the residue operation is that it gives one a relative measure of intensity. Our perception of a grey scale image is often in terms of areas of *brightness* (ie. peak) and *darkness* (ie. valley). The residue operations thereby provides a means to identify these peaks and valleys.

3.6 Thresholding

Thresholding is used in various stages during processing. Instead of using a fixed threshold, the threshold value in each case is derived from only the relevant part of the image using *strength percentile*. Strength can either be the result of an edge detection operation or a morphological filtering step. Percentile in each case is defined in reference to some minimum strength so that only pixels with at least minimum strength are considered. Our definition of percentile is different from the conventional. A percentile of x for strength y will indicate that x% of all relevant pixels have strength y and above. For example, in a well contrasted image, an edge with strength 20 may have a percentile of 45. That is 45% of all relevant edge pixels have a strength of 20 or higher. On the other hand, given the same strength in a poorly contrasted image where most edge pixels have very low strength, it may have a percentile of 15. Therefore, a strong (relatively speaking) pixel will have a very low percentile value, and vise versa. This shall provide us with a relative rather than an absolute measure which will automatically adapt to the quality of the image section in question.

3.7 Geometric Primitive Generation

In computer vision, it is often necessary to generate geometric primitive, such as lines, circles, ellipses, and parabolas. This is not only for display purposes, but also for parameterized model matching. The frequency with which this is

performed motivates a more efficient implementation rather than the straight forward analytic equation evaluation. The approach used here is adopted from a well-known technique, known as the Midpoint technique from computer graphics. [FvFH90] [Pit67] [Ake84].

The Midpoint technique is so-called because pixel coordinates are generated by examining the sign of the analytic equation evaluated at the midpoint between two neighboring pixels. A general analytic curve expressed as:

$$F(x, y) = 0 \tag{3.3}$$

describes a set of points along a curve which will satisfy the equation exactly. Yet, points which lie on either side of this curve will have opposite signs. Therefore, the sign at the midpoint will serve to indicate on which side of the midpoint the curve in fact passes through, and therefore the neighboring pixel on that side should be included in the digitized curve.

However, this still requires the function evaluation at the midpoint, which will not be much of a saving. The saving comes from the fact that the function evaluation can be done incrementally, ie. the function value at $midpoint_1$ can be calculated by adding a suitable amount to that at $midpoint_0$. The amount to be added shall vary depending on whether a square or diagonal move is performed. The square and diagonal move corresponds to whether only one or both coordinates are changed during the move. If the degree of the analytic equation is higher than one (ie. not a line), these update amounts will not be constant, but they can be calculated again incrementally just like the function itself by keeping track of their changes depending on whether a square or diagonal move is performed. Therefore, the bulk of the calculations can be reduced to additions rather than multiplications. Furthermore, providing that all the coefficients in the analytic equation are integers, all these calculations can be done via integer rather than floating point arithmetic. These two improvements translate into tremendous saving in terms of execution time requirements.

Therefore, all we need is to determine the initial function value at the midpoint, d, the update term for a square move, u, the update term for a diagonal move, v, and perhaps the changes of the update terms themselves upon a square or diagonal

move, δu_{square}, $\delta u_{diagonal}$, δv_{square}, $\delta v_{diagonal}$. The detailed development of these terms for line, circle, and general conic can be found in [FvFH90]. Although lines and circles can both be represented as special cases of conics, they are developed separately to take advantage of the lower degree of complexity and symmetry. Based on these, modified algorithms are developed for generating even and odd diameter circles, circle arcs and parabolic curves. Their derivations are detailed in appendices A, B and C.

3.8 Function Optimization

Deformable template modelling is employed by two modules in this system. One of the key components of this technique is function optimization. The task of finding the best set of parameters for a particular template is equivalent to locating the optimal point within the parameter space as defined by the template energy equation[3].

Yuille *et al.* used a modified version of the *Gradient Descent* method [YCH88]. The idea is to evaluate not only the function at each step but also the partial derivatives as well to determine the instantaneous gradient. The update direction and step size are determined by the angle and magnitude of this instantaneous gradient. There are two problems associated with this method. Firstly, the gradient is only an indication of the direction of greatest change. Its magnitude is by no means an indication of how far the minimum shall lie. Therefore, more strictly speaking, a one dimension optimization should be performed along the gradient direction, rather than simply taking the gradient magnitude as the update step size. Secondly, and perhaps more importantly, the calculation of partial derivatives at every point is an expensive task. The difficulties lie in the fact that the analytic curve involved may be oriented in any direction, therefore the image coordinates (x, y) at every point along the curve must be compensated by an extra rotational transform before being used in the derivative calculation. Unless careful bookkeeping is done, this can amount to a very expensive task whose computational effort can easily surpass that of several simple function evaluations.

[3]See section 2.6 for a description of deformable template modelling.

In order to avoid the problems mentioned above, Craw *et al.* adopted the *Simulated Annealing* technique instead [CTB92]. This technique is so-called because it attempts to imitate the annealing process in metal processing. Unfortunately, since not all of us are material science engineers, the analogy though correct, often serves to mystify rather than clarify the concept. Therefore, disregarding the analogy, simulated annealing is probably better termed as probablistic optimization. The optimization process is divided into stages of decreasing probability of accepting a non-optimal point[4]. During any one stage, the currently accepted point is modified slightly to generate a new point which will be evaluated. Depending on the current probability of acceptance associated with that particular function value, and the outcome of a random *dice roll*, this point can be either accepted or rejected. Once enough points are accepted or a fixed number of function evaluations have been performed, it advances with the best point so far to the next stage which will have a lower overall probability of acceptance. The whole process is repeated until the last stage is done. Because of the random element involved, this method should be better at finding a global rather than local optimal. Furthermore, since there is no gradient involved, the step at each point is merely a function evaluation and should be fast. However, because it is based upon probability, a relatively large sample size is needed to ensure the probablistic trend does indeed hold. Hence, this method is often reserved for combinatorial problems with large parameter space.

In our case, care has been taken to limit the search space as much as possible through the context module. Therefore, it's hoped that a less elaborate technique can be used. The multidimensional optimization technique we chose to use is the Downhill Simplex method by Nelder and Mead [NR65]. A simplex is the simplest geometric object beyond that of a hyperplane. With an n-dimensional space, a simplex is composed of $n+1$ vertices and all the lines and planar surfaces formed among them. Each vertex has its corresponding function value. The idea is to optimize their collective value by modifying one or more of the vertices of the current configuration. The four possible modification steps are as follows, listed

[4]A point in the n-dimensional parameter space simply refers to a particular set of parameter values.

in the order in which they are executed:

1. Modify the point with the worst value by reflection. The point is projected across onto the opposite side of the hyperplane formed by the other vertices. The amount of projection is such that the distance between the point and the hyperplane is preserved.

2. If the above step achieves a new best point, attempt further improvement by expanding it even further along the same direction.

3. If step 1 fails to improve the configuration, re-modify the low point by contraction so that the point is pulled closer to the hyperplane.

4. If all fails, modify all vertices except the point with the best value by contracting them all toward the best point.

In the continuous case, the configuration is continuously modified, until the fractional difference[5] is smaller than some prescribed value. However, for practical purposes, we have taken the multidimensional space to be discrete rather than continuous. Therefore, the process will terminate when the same configuration is repeated in the multidimensional contraction step 4 listed above.

In order to use this method, we also need an initial simplex of $n+1$ vertices. These vertices can be taken simply as the origin, P_0, and n other points, P_j's as follows:

$$P_j = P_0 + \lambda e_j$$

where e_j is the unit vector along the j^{th} dimension and λ is usually a constant factor. Since Downhill Simplex is a greedy algorithm and is not guaranteed to find global optimum, some modifications are needed to improve the probability that a *good* solution is obtained. In our implementation, we require the algorithm to repeat the whole optimization process with the optimal point so-far being one of the $n+1$ vertices until the same optimum is reached twice in a row. Furthermore,

[5]Fractional difference is the sum of differences of function value between the best point and the others divided by the average value

to ensure more points are examined in consecutive runs, we use a random λ to generate initial simplex vertices. In addition, we also modify the vertex generation equation to include directions other than those of the unit vectors. In general, P_j of the k^{th} run can be generated as:

$$P_j = P_0 + \sum_{i=j}^{j+k} \lambda e_l$$

where $l = (i-1) \bmod n + 1$.

In the case of a single parabola, there are five parameters: the x and y coordinates of the center, the height a, the half-width b, and tilt angle θ[6]. Our problem domain is therefore a 5-dimensional space centered within the allowable ranges of these parameters. The only remaining requirement is a function by which each point can be evaluated. This is simply the energy function associated with each deformable template. It could be as simple as evaluating the average edge strength along the curve, or as complicated as calculating the average peak value of the area bounded by a number of different curves.

Since this optimization method involves only a fixed number of modification steps and function evaluations, but not derivative evaluations, it should be relatively easy to implement and quick to execute.

3.9 Elastic Spring Functions

In addition to function optimization, the application of deformable template modelling requires a means to formulate and quantitatively evaluate geometric relationships such as relative positions and proportional sizes among components in the template. Elastic spring functions are frequently employed for such a purpose [YCH88] [SW91] [CTB92]. Often it is desirable to have a rapidly increasing cost function so as to discourage deviation from the ideal state. This can be expressed as a quadratic spring function and is what we term a *SimpleSpring* function:

$$SimpleSpring(k, l_0, l) = k(l - l_0)^2 \tag{3.4}$$

[6]See appendix C for precise definitions of these terms

where l and l_0 are respectively the springs actual and ideal undeformed lengths. Therefore, the $(l - l_0)$ is the amount of compression or stretch, ie. deviation from the ideal state. This deviation is weighted by the spring constant, k, so that a high k represents a very stiff spring encurring a high cost per unit of $(deviation)^2$.

A more complex relationship can be modelled by treating compression and stretch differently. In addition, we expand the ideal state from a single point l_0 to a range $[l_1 \ldots l_2]$. One can imagine it as two springs anchored together back to back. We call this a *DoubleSpring* cost function and it is formulated mathematically as follows:

$$DoubleSpring(k_1, k_2, l_1, l_2, l) = \begin{cases} k_1(l_1 - l)^2 & \text{if } (l < l_1) \\ 0 & \text{if } (l_1 \leq l \leq l_2) \\ k_2(l - l_2)^2 & \text{if } (l_2 < l) \end{cases} \tag{3.5}$$

where l_1 and l_2 define the lower and upper bounds of the ideal range, and k_1 and k_2 are the spring constants on either end. We shall use the notation $SimpleSpring(k, l_0, l)$ and $DoubleSpring(k_1, k_2, l_1, l_2, l)$, to refer to the Simple and Double Spring functions as in equations 3.4 and 3.5. Figure 3.7 illustrates an example function of each.

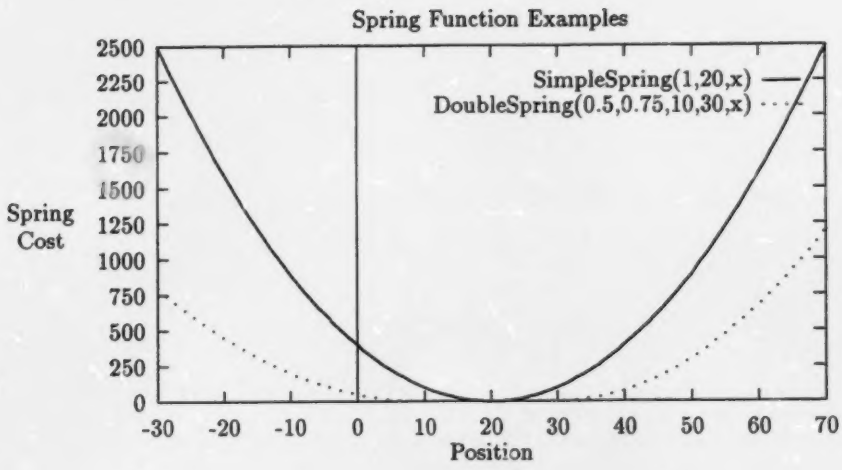


Figure 3.7: Spring Function Examples

Chapter 4

Context Module

The design of the facial context module is based upon the observation that though the sizes and distance among facial features may vary, their overall spatial arrangement remains the same. Consequently, any collection of objects with the proper spatial arrangement is potentially a face. A facial context is therefore simply a collection of distinct regions whose spatial arrangement resembles that of the eyes, eyebrows and mouth. The approach here is to first identify all distinct regions and then attempt to group these regions into plausible contexts. Depending on how closely they resemble the desired arrangement, they are ranked in terms of the likelihood of being the target face. The eventual objective is to obtain a list of potential face contexts ordered by face likelihood. Their validity will be checked in the confirmation step in order of decreasing likelihood until one is confirmed as the *real* face.

4.1 Image Segmentation

Image segmentation is the process of segmenting or dividing an image into distinct regions of similar characteristics. Since there is a separate confirmation step for verification and refinement, the emphasis here will not be in precision. Rather, we just want to locate regions of interest roughly and quickly. Figure 4.1 illustrates the segmentation flowchart and each process block is detailed in the following subsections.

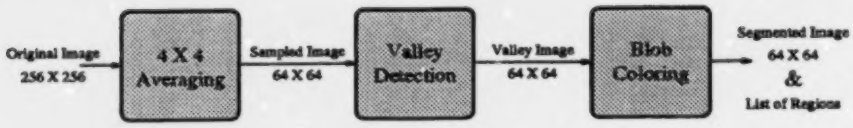


Figure 4.1: Segmentation Step Flow Chart

4.1.1 Resolution Reduction

Objects can often be more easily recognized in images that have a very low resolution. This is because the reduction in resolution typically reduces the amount of confusing details that may otherwise be present in its higher resolution counterpart. Experiments have shown that human subjects can identify a face from an image with a spatial resolution as small as 32×32 [Sam91]. An additional advantage of using a low resolution image is the reduction in computational requirement since it is often achieved by subsampling the original image and thereby reducing the dimensionality of the problem domain. Here, a small low resolution image is extracted from the input image by taking the average value among pixels within a 4×4 pixel region. Given an original image of 256×256, the extracted image will be of 64×64 and all subsequent segmentation processing is performed upon this low resolution image.

4.1.2 Valley Detection

The primary regions we wish to extract are those of the eyes, eyebrows, and mouth. Since they should appear as relatively darker objects than their respective surroundings, the morphological opening residue operation described in section 3.5 can be employed. The task on hand then is to extract all intensity valleys in the sampled image using the following 5×5 circle mask as shown in Figure 4.2. The choice of using a 5×5 circle mask is resolution dependent. If the input image to this stage is of a different size, a smaller or larger mask might have to be employed.

0	1	1	1	0
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
0	1	1	1	0

Figure 4.2: 5 x 5 Circle Mask

4.1.3 Blob Coloring

The valley detection technique discussed in the previous section detects intensity discontinuities. It must be followed with a linking procedure to assemble the detected pixels into a meaningful set of objects. Blob coloring is employed here for this linking task.

Blob coloring assigns a color to each distinct region in the picture. A distinct region is a set of *valid* pixels which are 8-connected neighbors. By *valid* pixels, we mean that these pixels have very high morphological opening residue values. In specific, a 15 percentile threshold is used to determine whether a pixel has a high enough value[1].

In practice, this process is done in two passes through the image, and is a modified version of that described in [BB82]. In the first pass, each point only has to examine half of its neighbors, which are on top or to the left (see Figure 4.3). If the center pixel X_c is connected to one of the four neighboring pixels, X_n as

X_N	X_N	X_N
X_N	X_C	

Figure 4.3: 8-connected Blob Coloring Mask

shown, it will be assigned the same color. If it is connected to more than one with different colors, then these colors will be marked equivalent, and the current pixel

[1]see section 3.6 for a detail explanation of percentile thresholding

is assigned one of these colors. Intuitively, this serves to identify regions which are initially divided but later joined together.

After the first pass is completed, the list of assigned colors can be compressed into a shorter list by merging colors which are marked equivalent, yielding a compressed color table. Then in the second pass, the original assigned colors in the image will be replaced by their true color in this compressed color table. Figure 4.4 illustrates the images generated during this segmentation step.

(a) Low Resolution Image (b) Valley Image (c) Segmented Image

Figure 4.4: Images Generated during Segmentation

In order to facilitate reasoning in the following facial context evaluation step, the segmented image is further reduced into a list of distinct regions with the following attributes:

- size
- length and width of the bounding rectangle
- average, maximum and minimum grey level
- centroid location

4.2 Context Constraint Model

It is hypothesized that eyes, being a pair of dark regions on a light surface (ie. skin), are the most striking features observed in a low resolution valley image. Therefore, each potential context should at minimum possess one such horizontal

pair. Based on their positions, locations of other valley regions corresponding to the eyebrows and mouth can be estimated. Finally potential contexts can be evaluated and ranked according to the results of the pairing and context completion processes.

4.2.1 Pair Analysis

The *horizontal pairness* of two regions is accessed in terms of their shape resemblance and position correspondence. Shape resemblance is measured by overlapping the bounding rectangle of the two and taking the proportion between their symmetric difference and intersection as shown in Figure 4.5:

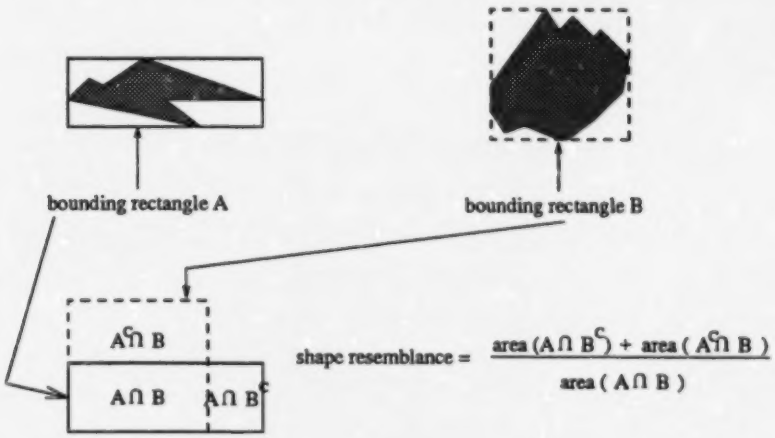
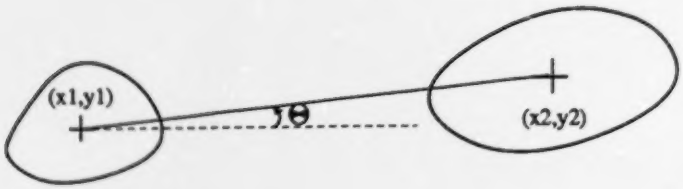


Figure 4.5: Shape Resemblance Measure

Positional correspondence is derived by forming a vector between the centroid of the two regions and measuring the amount of deviation of this vector from the horizon (see Figure 4.6). Therefore, position is measured in radian and is 0 when the pair is exactly horizontal.

In order to obtain a valid pair, both of these measures must pass their respective thresholds. The shape threshold is 3.0 (ie. shape resemblance < 3.0) and the position threshold is 15° (ie. position correspondence $< \frac{\pi}{12}$). Contexts are formed from region pairs which satisfy these two threshold conditions.



$$\text{Positional Correspondence} = \tan^{-1}\left(\frac{y1 - y2}{x2 - x1}\right)$$

Figure 4.6: Position Correspondence Measure

Since it is very likely that the head may have a slight tilt, the emphasis of the overall ranking of all valid pairs is placed on shape rather than position. The following heuristic is found to yield satisfactory results.

$$COST_{pair} = 10 * shape^2 * \frac{position}{PI/18} \tag{4.1}$$

Note that the position measure is normalized with a 10° angle, the shape measure is squared to emphasize its importance, and the coefficient 10 is added to increase the resolution of the integer $COST_{pair}$ measure.

4.2.2 Context Completion

Once the eye regions are identified, the module then attempts to complete the context by locating *plausible* regions positioned at the estimated locations. Figure 4.7 illustrates the search regions for the eyebrows and mouth. All measurements are expressed in reference to the distance between the two eye regions' centroids (ie. ref). The template component is marked found if any pixels within its corresponding search region is found to belong to a *wide region* defined as follows:

$$wide = \begin{cases} 1 & \text{if } (\frac{height}{wide} > 0.5) \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

The total cost of missing components is simply the sum of all the missing parts:

$$BROW = \begin{cases} 0 & \text{if eyebrow exists as } wide \text{ region} \\ 20 & \text{otherwise} \end{cases}$$

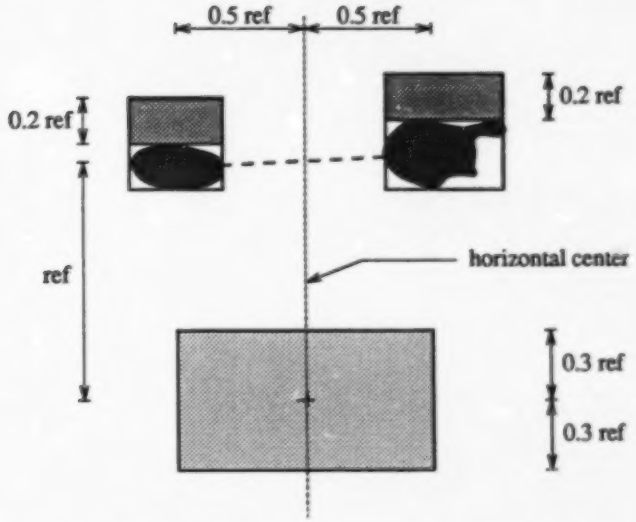


Figure 4.7: Context Component Search Regions

$$MOUTH = \begin{cases} 0 & \text{if mouth region exists as } \textit{wide} \text{ region} \\ 10 & \text{otherwise} \end{cases}$$
$$COST_{missing} = BROW_{left} + BROW_{right} + MOUTH \quad (4.3)$$

Notice that the brows are assigned a higher weight, since the search area for them is much more limited. Therefore, if a wide region is found, it tends to be a more reliable measure.

4.2.3 Context Evaluation

The results of the two previous cost measures are combined to form an overall context evaluation measure:

$$COST_{context} = COST_{pair} + COST_{missing} \quad (4.4)$$

where $COST_{pair}$ and $COST_{missing}$ are as defined in equations 4.1 and 4.3.

The above cost measure is a heuristic derived through conservative experiments. It can not guarantee that the correct context will be ranked first in the

list, but it will certainly include it in the list for further confirmation. The tuning and improvement of this cost measure will reduce the time requirement.

4.3 Experimental Results and Discussion

The context module described has been implemented and tested with a set of 67 images, all of which are without spectacles. In order to assess whether a context is correct, bounding boxes are put around regions which are to be processed furthered by the confirmation modules. These are namely the eye and the mouth regions. The eye regions are defined by the bounding rectangles around the initial region pair which form the basis of each context. The mouth region, on the other hand, is more difficult since it is often missing or fragmented after segmentation. Therefore, we opt for a simple estimate approach based on the distance of separation between centroids of the two eye regions. Figure 4.8 illustrates how this mouth region is formed. The correctness of each context is simply judged

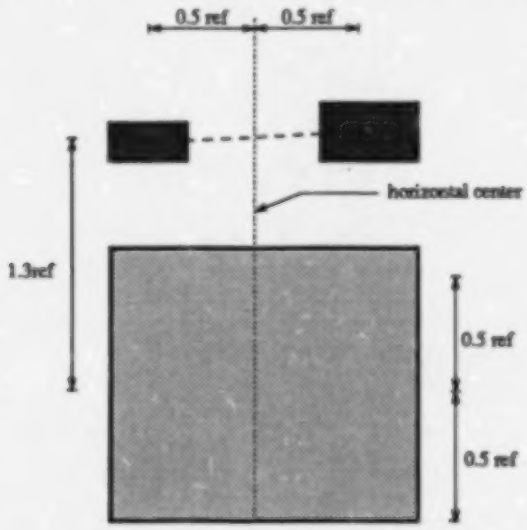


Figure 4.8: Module Region Extraction

by whether the eyes and mouth are included in their corresponding estimated position. Figure 4.9 illustrates such example contexts. Although the estimate

(a) Correct Context (b) Wrong Context

Figure 4.9: Example Contexts

may seem to be over-generous, it is kept conservative in order to account for the potential error in locating the centroids of the eye regions.

The actual run time ranged from a low of 0.010 sec to a high of 2.117 sec. The average run time was 0.863 sec with a 0.604 sec standard deviation. Of the 67 test images, the context module correctly identified the real face as being one of the potential contexts in 64 images. Fifty-seven of these ranked it as being the first in the list, while the remaining 7 ranked it within the top 3 potential contexts. Of the 3 that failed, lighting was generally insufficient (see example in Figure 4.10). As a result, the segmentation step was unable to pick up one or both of the eye regions. It was expected that if pictures were of better quality, better performance could be achieved with no modification to the module. The result is very promising since it places the module at a 96% hit rate with minimal execution time.

In addition, a second set of 10 images of bespectacled individuals were tested. The system failed on all but two. It failed because the segmentation step was unable to cope with the presence of spectacle. Either one or both eyes were often included in the same region as that of the eyeglasses. For the two that succeeded,

(a) Low Resolution Image (b) Valley Image (c) Segmented Image

Figure 4.10: Failed Context

due to reflectance off the glass plane, the resolution on the eye region was minimal. It is doubtful that the eye module would be able to pick up the eye shape reliably. The result is disappointing though not unexpected since be-spectacled cases are really beyond the scope of the current design. However, given the large number of be-spectacled individuals among today's population, it indicates that there is a need for an alternative context module before this system can be applied in real practice. This alternative context module must either take glasses into account (ie. a bespectacled face context) or adopt a completely different strategy which is not influenced by the presence of glasses. One possible approach is the use of gradient angle direction to find noses as described in [RS92]. Robertson reported a success rate of 95% for a test set of 16 images. It would be interesting to see how successful it was among only bespectacled cases in order to decide whether it would complement our current context module.

Chapter 5

Eye Module

The objective of the eye module is to confirm and refine the eye location in the given region. The most widely used approach in the past had been approximating the eyes with intensity minima [CEL87] [CJCM86] [Lam87]. This was often accompanied with other constraints, such as location of other previously detected features. The problem of using this simple approach is two-fold. First, image charasteristics, such as deep valley or high intensity changes, could be results of many objects other than the eyes, especially if the background is non-uniform. They may be used to establish possible locations of eyes, as we had in the context module. However, it is far from being adequate as a confirmation guideline. Therefore, a more confident shape measure based on a high level modelling paradigm, such as the Hough transform [Nix85] or deformable template modelling [YCH88], has to be employed.

Instead of using the deformable template technique for the entire model, we opt for a two-step approach based on the use of Hough transform and deformable template. In the first step, we shall attempt to locate the iris modelled as a circle using classical Hough transform technique, as in [Nix85]. This is essentially equivalent to the first three epoches in [YCH88] where the energy function is tuned for attracting the template to the iris. After the irises have been located, a template of only the bounding parabolas is used to correctly orient and complete the description of the eye.

The hope is that the first step will guarantee a more exhaustive and global

search with the least expensive template element over the entire given region to locate the most likely location for the eye. Therefore, it should overstep any possible local minima and reduce the parameter space, both of which are important factors that may influence the correct placement of the final template during the later minimization process.

The detailed descriptions of these two steps are covered in the remainder of this chapter, followed by their experimental results and discussion.

5.1 Circle Hough Transform

In this section, we shall examine how the Hough transform technique can be employed to extract the irises. A general discussion of Hough transform can be found in section 2.5. The discussion here is limited to that of finding dark circles in the subregion derived from the current context (as described in section 4.2 of the context module).

The parametric equation for circle used is the standard form as shown:

$$(x-a)^2 + (y-b)^2 = r^2$$

It has three parameters: the center of the circle (a, b) and the radius r. Therefore, the parametric space is a three dimensional array with axis a, b, and r. Classical Hough transform technique prescribes that for every edge point found, calculate its corresponding circle center for every possible radius value.

Unfortunately, there are two problems in applying this algorithm in practice. First, the parametric equation of a circle is an equation with two second degree parameters. The inverse solution, requiring both square and square root evaluation, is inherently expensive. This is especially true since it has to be performed for each edge point in the image over the range of all allowable radii. Second, since real images are used as input, the edge information input to the Hough circle module is going to have a certain amount of error. Consequently, the parameter space will have regions of slightly higher accumulated count rather than a sharp spike. This creates a difficulty in distinguishing good circles from the rest.

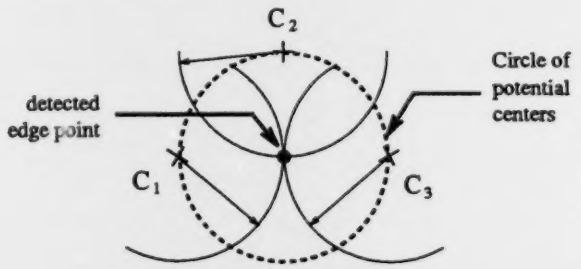
In order to solve these problems, the classical circle Hough transform has to be amended by the following means:

- Incorporating gradient direction information
- Applying pair constraint to circles found

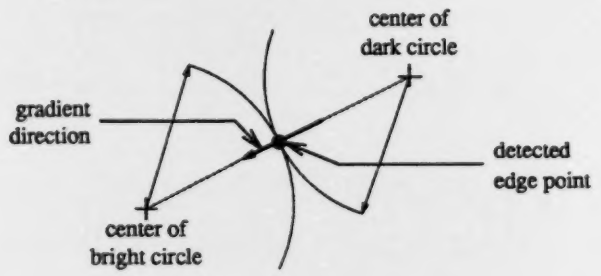
The first item is aimed at reducing the number of points generated by the Hough circle module, whereas the last one is aimed at resolving the circle selection problem. The application of these two amendments are described in the following subsections.

5.1.1 Parameter Space Generation

If gradient direction is not known, the edge point can lie anywhere on a circle even if the radius R is fixed. Therefore, the center of the circle can be anywhere a distance of R away as shown:



If we know the gradient direction, there can only be precisely two possible circles for each fixed radius. One of these can be further eliminated depending on whether a dark or light circle is to be located.



However, the gradient direction is not precise. Therefore, instead of having only one circle, we will have a number of them for each radius, R. Their centers

will lie on an arc of R distance away from the edge point. The angular length of this arc depends upon the accuracy of the edge detector. We use two 3×3 Sobel masks to calculate the horizontal and vertical edge components, and a 3×3 Laplacian mask to determine the edge locations. Edges are detected as described in section 3.4 using the Sobel and Laplacian kernels. The edge pixels are further thresholded with a 60-percentile magnitude requirement. Since there are only 8 distinct directions, the minimum error will be at least ± 22.5 degree. In practice, we allow for a small safety margin and use a ± 30 degree arc to generate the centers of candidate circles for each radius. As well, A modified midpoint circle algorithm is used to generate the arcs, and they are explained in appendices A and B.

Through detailed inspection of all test images, it was decided that the iris diameter ranges from 7 to 14 pixels. That is to say that an iris of diameter less than 7 pixels is too small for robust detail analysis, while an iris of diameter greater than 14 pixels will mean that the face is too large to be accommodated entirely in the image. This of course is image size dependent and is valid for the case of a 256×256 image only. This range is used in the circle Hough transform step to limit the search to a reasonable range. Although this still involves a certain amount of computation in calculating the circle parameter, it is a substantial saving over that without use of gradient angle.

To further this saving, we formulate our calculation in a systematic and hierarchical manner, so that some of the intermediate results can be saved and reused. In general, the error in gradient direction is ± 30 degree. Since, the gradient direction for each edge point is known at the onset, the starting and ending angles of the arc span will remain fixed regardless of the radius. Therefore, the extreme endpoints of the arc can be calculated incrementally using additions only. Multiplication and trigonometric functions, both of which are expensive operations, are only evaluated once at the beginning of the loop to reduce computational burden.

5.1.2 Optimal Circle Selection

Local Averaging

In addition to error in gradient angle, often edge location is subjected to error. Consequently, the computed center can be slightly off from the true center. Ideally, we should compute the center from all possible locations within the margin of error, but in practice, this could easily quadruple the amount of computations or worse. Therefore, instead of moving the edge pixel around, we perturb the computed center location slightly to achieve the same effect. This is equivalent of performing a local averaging operation over the $a - b$ plane within the parameter space. This step will be executed before actual selection is contemplated.

Preliminary Screening

In general, because the parameter space may be arbitrarily huge, exhaustive examination can be computationally prohibited. Therefore, it is advantageous to perform some preliminary screening to weed out very poor cases. It was expected that the parameter space is a very sparse matrix with the majority of the cells having a very low accumulated count. A rough screening can be applied using statistical measures. For each diameter R, we calculate the standard deviation σ and mean μ of accumulated counts. The cutoff is taken to be the sum of the two. Therefore, only cells with accumulated count greater than $\sigma + \mu$ are screened through to be examined in detail.

Circle Evaluation

Theoretically, comparison among circles can be based solely upon the accumulated count. However, a circle with a larger diameter has a longer contour, and thus will inherently have a higher count than that with a smaller diameter. On the other hand, if we normalize the count by dividing it with the diameter, it will be equivalent to measuring the percentage of circumference covered. Yet, in this case, circles with very small diameter can easily achieve a high percentage just from a few accidentally aligned edge points. Through experimentation, the following

equation seems to achieve the proper balance between the two extremes:

$$SCORE_{circle} = count * \frac{count}{diameter} \tag{5.1}$$

The first term rates the circle by the accumulated count of edge points. The second term evaluates it by coverage percentage.

Circle Pair Constraint

However, the combination of edge inaccuracy and accidental alignment of edge points produces a *blurred* parameter space. The distinction between good and bad circles can no longer be identified by a clear sharp spike as predicted by theory. To overcome this problem, we apply an additional constraint in circle selection, namely a pair of identically sized circles, one in each eye region, must be located. The circle pair is evaluated using the following equation:

$$SCORE_{pair} = SCORE_{circle1} + SCORE_{circle2} - tilt^2 \tag{5.2}$$

where $SCORE_{circle1}$ and $SCORE_{circle2}$ are the score computed from equation 5.1 for the two circles, and *tilt* is the angular measure in radians between the centers of the two circles. Equation 5.2 is used in a ranking step to obtain an ordered list of potential iris pairs. In the absence of a better measure for the time being, only the first pair is used in the deformable template step. This can be refined to an iterative search down the potential iris list, once a reliable measure has been established to distinguish a good fit from a bad.

The processing region to be used by the deformable template step is estimated using the iris position and size as shown in Figure 5.1. The revision of bounding box size and location is necessary because the initial one provided by the context module is only an extremely crude estimate. Not only that it may contain unrelated and potentially distracting objects, it may also not have included the entire eye to begin with.

5.2 Deformable Template for Eye Boundary

The eye boundary template is composed of only two parabolic curves as illustrated in Figure 5.2. The energy function used is based only on edges and internal con-

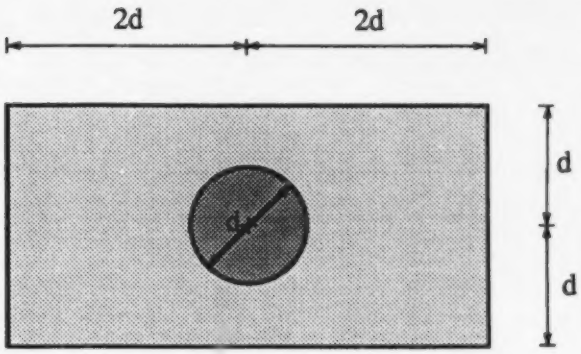
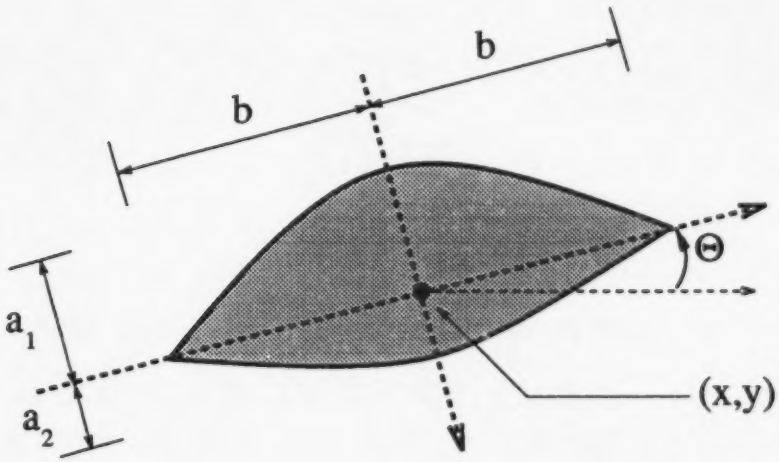


Figure 5.1: Bounding Box for the Eye Boundary Step



Legend :

- Θ - the tilt angle
- (x,y) - the coordinates of the parabola center
- b - the parabola halfwidth
- a_1 - the upper parabola height
- a_2 - the lower parabola height

Figure 5.2: Eye Boundary Template

straint force. This is because by the time this template is applied, the rough location has already been extracted and therefore there is no need for the valley and peak constraints as required in the original formulation by Yuille *et al.* [YCH88].

Ideally, one should be able to locate each eye boundary separately. However, a solution achieved through numerical optimization is not guaranteed to be globally optimal. Coupling this with a noisy image, the parameterized template found may fluctuate a great deal. Rather than imposing a pro-processing measure to amend any discrepancy between the shape found for the two eyes, it will be far easier to optimize the two together. The final template used is hence composed of two instances of that shown in Figure 5.2. The energy function of our combined template can be divided into five separate pieces: the upper and lower parabolas for the two eyes, and the combined shape function.

The upper parabola edge functions are defined to be the average edge strength over the boundary of the upper parabola for each eye:

$$E_{leftupper} = \frac{1}{LeftUpperLength} \int_{LeftUpperParabola} \Phi_e(\vec{x}) ds \tag{5.3}$$

$$E_{rightupper} = \frac{1}{RightUpperLength} \int_{RightUpperParabola} \Phi_e(\vec{x}) ds \tag{5.4}$$

The lower parabola edge functions are defined similarly:

$$E_{leftlower} = \frac{1}{LeftLowerLength} \int_{LeftLowerParabola} \Phi_e(\vec{x}) ds \tag{5.5}$$

$$E_{rightlower} = \frac{1}{RightLowerLength} \int_{RightLowerParabola} \Phi_e(\vec{x}) ds \tag{5.6}$$

The shape function, however, is more complex:

$$\begin{aligned} E_{shape} = {} & X_{left} + Y_{left} + B_{left} + A1_{left} + A2_{left} + \\ & X_{right} + Y_{right} + B_{right} + A1_{right} + A2_{right} + \\ & B_{pair} + A1_{pair} + A2_{pair} + \theta pair \end{aligned} \tag{5.7}$$

The first and second five terms of the above equation control the shape of each of the eyes, and the last five terms shall encourage symmetry between the two. All of these are to be expressed using either *SimpleSpring* or *DoubleSpring* functions as described in section 3.9. The spring constants and ideal states, as well as

Shape Term	Variable l	Spring Constants k_1	Spring Constants k_2	Ideal State l_1	Ideal State l_2	Range from	Range to
X_{left}	$X(left)$	1.0	-	$X(left_{iris})$	-	$X(left_{iris})-$ $DIA(iris) * 0.5$	$X(left_{iris})+$ $DIA(iris) * 0.5$
Y_{left}	$Y(left)$	4.0	0.5	$Y(left_{iris})+$ $DIA(iris) * 0.1$	$Y(left_{iris})+$ $DIA(iris) * 0.2$	$Y(left_{iris})-$ $DIA(iris) * 0.1$	$Y(left_{iris})+$ $DIA(iris) * 0.5$
B_{left}	$B(left)$	4.0	0.5	$DIA(iris) * 1.2$	$DIA(iris) * 1.5$	$DIA(iris) * 0.8$	$DIA(iris) * 3.0$
$A1_{left}$	$A1(left)$	1.0	2.0	$B(left) * 0.2$	$B(left) * 0.5$	$DIA(iris) * 0.6$	$DIA(iris) * 1.0$
$A2_{left}$	$A2(left)$	1.0	2.0	$B(left) * 0.05$	$B(left) * 0.3$	$DIA(iris) * 0.2$	$DIA(iris) * 0.5$
X_{right}	$X(right)$	1.0	-	$X(right_{iris})$	-	$X(right_{iris})-$ $DIA(iris) * 0.5$	$X(right_{iris})+$ $DIA(iris) * 0.5$
Y_{right}	$Y(right)$	4.0	0.5	$Y(right_{iris})+$ $DIA(iris) * 0.1$	$Y(right_{iris})+$ $DIA(iris) * 0.2$	$Y(right_{iris})-$ $DIA(iris) * 0.1$	$Y(right_{iris})+$ $DIA(iris) * 0.5$
B_{right}	$B(right)$	4.0	0.5	$DIA(iris) * 1.2$	$DIA(iris) * 1.5$	$DIA(iris) * 0.8$	$DIA(iris) * 3.0$
$A1_{right}$	$A1(right)$	1.0	2.0	$B(right) * 0.2$	$B(right) * 0.5$	$DIA(iris) * 0.6$	$DIA(iris) * 1.0$
$A2_{right}$	$A2(right)$	1.0	2.0	$B(right) * 0.05$	$B(right) * 0.3$	$DIA(iris) * 0.2$	$DIA(iris) * 0.5$
B_{pair}	$B(right)$	4.0	-	$B(left)$	-	-	-
$A1_{pair}$	$A1(right)$	2.0	-	$A1(left)$	-	-	-
$A2_{pair}$	$A2(right)$	4.0	-	$A2(left)$	-	-	-
$\theta pair$	$i\theta(right)$	0.5	-	$-\theta(left)$	-	-	-

where $N(left)$ is the N parameter of the left eye boundary;
$N(right)$ is the N parameter of the right eye boundary;
$N(left_{iris})$ is the N coordinate of the left iris;
$N(right_{iris})$ is the N coordinate of the right iris;
$DIA(iris)$ is the diameter of the iris pair;

Table 5.1: Eye Boundary Shape Function Coefficients

the allowable range for each, are derived mostly through experimentations and are summarized in Table 5.1. To conserve space, the *SimpleSpring* function is presented as a subset of the *DoubleSpring* function with k_1 and l_1 substituted in as k and l_0.

The evaluation and optimization of eye template are done using the Midpoint technique and Downhill Simplex respectively. Their descriptions can be found in Chapter 3.

5.3 Experimental Results and Discussion

We recall that the objective of a feature inspector is two-fold. First, given the correct subregion, it must be able to refine the feature location and describe

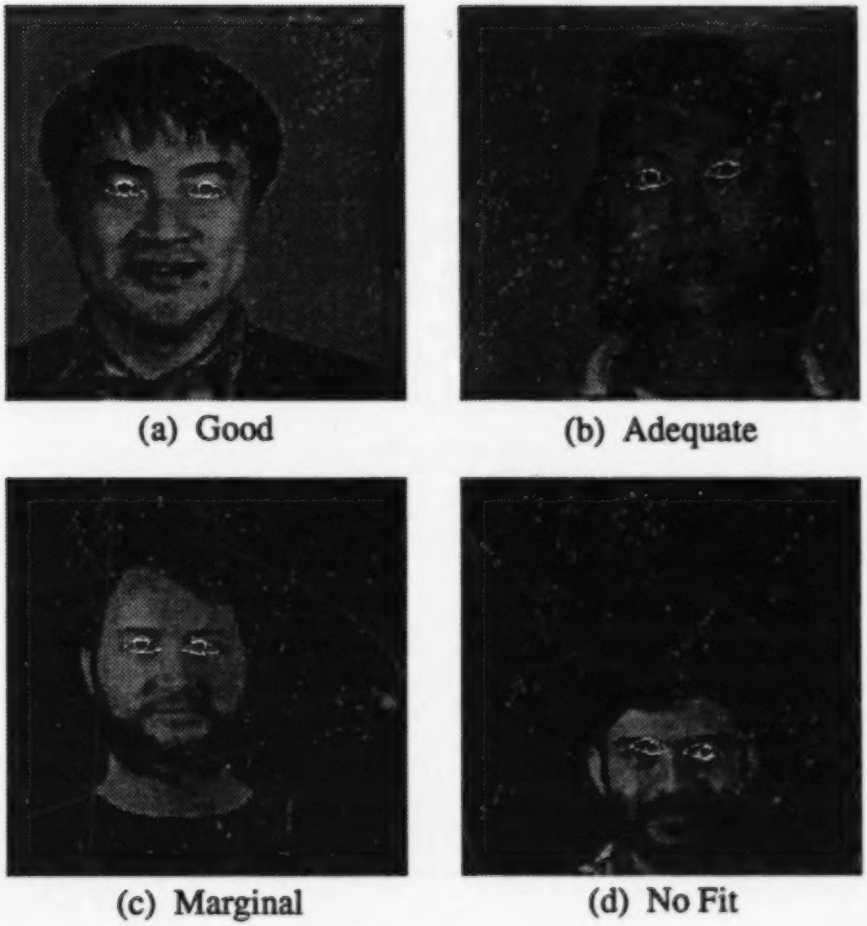
Rating	Number	Percentage	Description
Good	19	29	well fitted
Adequate	35	55	resaonable fit with center at the correct position.
Marginal	6	10	some error in fit.
No Fit	4	6	failed completely to lock onto one or both eyes.

Table 5.2: Eye Module Results

its shape in terms of the geometric model used. Second, it should be able to determine whether the best case found does indeed constitute a plausible instance of the feature in question. Hence, test runs were set up to investigate these two issues.

The first investigation was aimed at testing the detection capability of the eye module. The test was performed on proper eye regions, selected from the 64 images that were correctly identified by the context module. The objective was to decide whether given the proper region, the eye module could correctly extract the eyes. The final template fit was overlaid on the original image and subjectively rated for goodness-of-fit. The results are classified into four rating categories similar to that used by Shackleton *et al.* [SW91]. The results of this run are summarized in Table 5.2. An example of each is shown in Figure 5.3

In general, it was found that an exact fit is difficult to achieve since the parabolic approximation is after all just an approximation, and it is difficult to conceive that most human eyes would in fact confirm to the parabolic shape as dictated. However, a relatively **Good** fit can be attained if the image is well contrasted and of a bigger size. Yet, many images which were taken early on in this investigation have tended to have lower picture quality and less effective use of available image area (ie. the head size is relatively small). This contributed to the high percentage in the **Adequate** category. In those cases, the edge information was less defined, therefore the fitted template might be slightly off at some places, though the overall fit remained fairly close.



(a) Good (b) Adequate

(c) Marginal (d) No Fit

Figure 5.3: Eye Template Fit

Among the **Marginal** and **No Fit** cases, the major source of error came from the iris size, an important reference measure for the deformable template step. This was partly due to the image quality and resolution problems mentioned earlier. A more important problem inherent to our module lies on the use of the circle Hough transform to locate the iris. In doing so we assume that the iris will appear as a circular region. However, for many individuals with small eyes (or more precisely, narrow eye openings), it is difficult to observe the circular iris since most of it is occluded. An obvious modification is to be rid of the Hough transform step and apply the deformable template technique using the complete template including the circular iris. This is essentially what was employed in [YCH88] [SW91] [CTB92]. As we mentioned in section 2.6, the success of this technique depends very much on how localized they can limit the area of processing. Yet, without the Hough transform step to place the template relatively close to the eyes and restrict the search space, we anticipate potential problems with local minima during the optimization step. The compromise is perhaps to retain the use of circle Hough transform as an initial estimate of where the eye is, and use the complete template rather than just the bounding parabolas in the deformable template step to fine-tune the shape and location of both the iris and eye boundary. This is however left for future investigation.

In the second investigation, we want to derive a quantitative measure to distinguish **Good** to **Adequate** fits from those that are **Marginal** to **No Fit**. In addition to the previous 64 eye regions, 46 additional were selected randomly from incorrectly identified contexts. These incorrect eye regions were examined in order to observe the behavior of our eye module under non-ideal conditions. The test samples were divided into two groups. One group was comprised of those which were ranked **Good** to **Adequate** from the initial set. The second group consisted of those which were ranked **Marginal** to **No Fit** from the initial set, and all of the additional incorrect eye regions. The total energy cost of the resultant template was examined in each and a histogram was made with discrete bins of 10 units apart. The results are as shown in Figure 5.4:

From Figure 5.4, we can observe that the template energy cost for the *Good* eyes is generally lower than that of the *Bad* instances. Unfortunately the distribu-

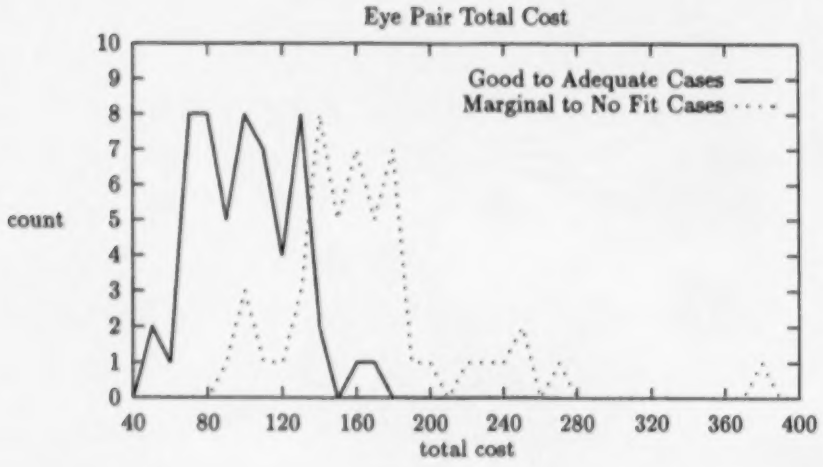


Figure 5.4: Eye Template Energy Cost Distribution

tion pattern between the two is not as distinct as we had hoped for. Upon careful examination of the handful of bad instances which managed to achieve relatively low template energy cost, we noticed that in most cases *eyes* were found on the eyebrows (see Figure 5.5). These eyebrows in fact provided fairly good fit for our Eye Boundary template which was defined using only bounding parabolas. One obvious improvement is of course to include the circular iris as we had suggested earlier. A second improvement would be to examine not just the total cost, but also the individual piece cost. The principle idea behind this is that with an incorrect instance, at least one of the eyelids or irises will be poorly fitted. However, the determination of the cutoff values for each piece is a difficult task. The validity of this idea still remains to be proven through further testing.

The runtime for all test images ranged between 7.580 sec and 20.650 sec. The average was 11.545 sec with a 3.003 sec standard deviation. This would seem to be a significant improvement over the reported runtime by Yuille *et al.* of 5 to 10 minutes. Both ours and Yuille's were tested on a Sun4 Unix machine. Our Sun4 is 1.5 times as fast as the low end Sun4. Hence, even if their experiment was done

Figure 5.5: Failed Eyes

on a low end Sun4, our algorithm is still at least 16 times as fast. This clearly indicates that our effort in reducing computational expense through efficient geometric primitive generation and numerical optimization is well worthwhile. As long as careful bookkeeping is maintained, the addition of the circular iris in this step should not impact the performance by too much.

Chapter 6

Mouth Module

The objective of the mouth module is to confirm and locate precisely where the mouth is within the given region. In the past, many researchers used simple models such as long thin valley, or simple horizontal edge. As we pointed out in the eye module, although a simple model can be used as an initial estimate, it is most likely insufficient as a reliable confirmation guideline. This indicates the need of a considerable more elaborate design than what has been done in the past.

The conceptual design of the mouth module parallels that of the eye module and is divided into two stages. The first stage uses the classical Hough transform technique to identify long horizontal edge segments within the subregion supplied by the context module. Based on this refined estimate, the second stage will employ the deformable template technique as proposed by Yuille to identify the precise location and shape of the mouth. The description and experimental results of each of these two stages are detailed in the remainder of this chapter.

6.1 Line Hough Transform

Extraction of line is by far the most well-established and well-used application of hough transform for two reasons. Firstly, line has a low dimensionality, making it computationally easy to manage in practice. Secondly, it is the original application for which Hough transform was designed for, hence it is well studied. The

formulation of our routine is primarily based on the approach established in past research. It uses the normal representation of line [1]:

$$y\sin(\theta) - x\cos(\theta) = \rho \tag{6.1}$$

where x, y are the Cartesian coordinates of a point along the line, and θ and ρ describe the orientation and length of its normal vector anchored at the origin. Figure 6.1 illustrates this relationship. This is a convenient representation because

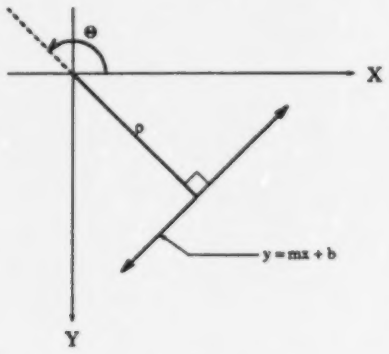


Figure 6.1: Normal Representation of Line

a gradient angle by definition is normal to the edge and therefore can be used directly to index into the appropriate parameter space elements.

6.1.1 Parameter Space Generation

The parameter space is a two dimensional one with axes θ and ρ. The gradient direction, θ, ranges from 85 to 95 since it only has to account for relatively horizontal lines. By translating the origin to the center of the image, ρ has to cover from $-\rho_{max}$ to $+\rho_{max}$, where ρ_{max} is half the length of the longest possible line in the given area, namely $0.5 * \sqrt{height^2 + width^2}$. Since we are interested in obtaining finite lines, each parameter space cell must keep track of the two extreme endpoints in addition to the accumulated count. Because of error in edge detection, a $\pm 30°$ error must be incorporated. That is to say that for each valid

[1] This representation is slightly different from the conventional because of the inverted Y axis used in our coordinates system.

edge point, a ρ is calculated for each angle between $\theta-30°$ and $\theta+30°$. Therefore, a handful of parameter space cells rather than a single one will be updated for each edge point. The valid edge points are extracted by edge detection techniques described in section 3.4 and then thresholded with a 60-percentile magnitude requirement.

6.1.2 Horizontal Line Selection

The selection process for the optimal horizontal line is roughly the same as that used in circle selection. First we perform local averaging within the parameter space cell. Then we apply the same preliminary screening criteria as described in section 5.1.2 to extract a list of reasonably *good* horizontal lines. These lines are then ranked using the following heuristic:

$$SCORE_{line} = length - 0.5 * (endpoint_1.y + endpoint_2.y) \tag{6.2}$$

The result of ranking by equation 6.2 is an ordered list of potential mouth lines. Ideally, they would be checked one at a time until a mouth is confirmed at that location. However, for now only the first horizontal line in the list is used for testing in the subsequent deformable template step. In addition, the region of processing is localized to only the immediate neighborhood surrounding this line as shown in Figure 6.2.

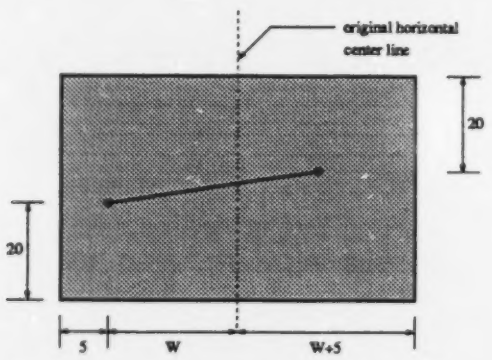
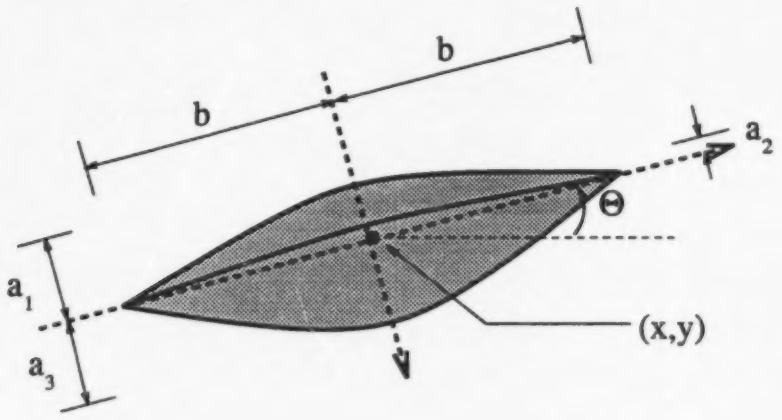


Figure 6.2: Bounding Box for the Mouth Deformable Template Step

6.2 Deformable Template for Mouth

For a detailed analysis of mouth shape and location, we employ the deformable template technique. The template we use to model the mouth consists of three parabolic curves as shown in Figure 6.3. It is essentially a simplified version of



Legend :

- Θ - the tilt angle
- (x,y)- the coordinates of the parabola center
- b - the parabola halfwidth
- a_1 - the upper parabola height
- a_2 - the middle parabola height
- a_3 - the lower parabola height

Figure 6.3: Mouth Template

that used in [YCH88]. The motivation is primarily computational: the simpler the template, the less computational effort is needed to evaluate it. Nevertheless, there is still enough structural constraint that it should be able to lock on the mouth just as well as that of the more complicated one in [YCH88].

Again, similar to the eye template, the mouth template is edge based only. Valley and peak forces are presumed unnecessary since the region within which this is to be located should be relatively localized. Therefore, the external energy

function is composed of only three separate terms corresponding to the edges of the three parabolas in the template:

$$E_{upper} = \frac{1}{UpperLength} \int_{UpperParabola} \Phi_e(\vec{x}) ds \tag{6.3}$$

$$E_{midline} = \frac{1}{MiddleLength} \int_{MiddleParabola} \Phi_e(\vec{x}) ds \tag{6.4}$$

$$E_{lower} = \frac{1}{LowerLength} \int_{LowerParabola} \Phi_e(\vec{x}) ds \tag{6.5}$$

where $\Phi_e(\vec{x})$ is the edge strength of the point $\vec{x}$ on the curve; E_{upper}, $E_{midline}$, E_{lower} are the energy functions for the upper, middle, and lower parabolas respectively.

The shape function controls only the size proportion among parameters to attain an average mouth shape,

$$\begin{aligned} E_{shape} = X_{mouth} &+ Y_{mouth} + B_{mouth} + A1_{mouth} + A2_{mouth} \\ &+ A3_{mouth} + \theta_{mouth} \end{aligned} \tag{6.6}$$

Terms in Equation 6.6 are expressed using SimpleSpring and DoubleSpring functions. Using the same notation as in section 5.2, the experimentally derived spring coefficients and range of operations are tabulated in Table 6.1.

One will notice that there is more tolerance placed upon the lower range of the $A1$ and $A3$ terms so as to allow them to degenerate into straight lines. This is designed primarily for accommodating people with very thin lips.

The generation of these geometric primitives are based upon the Midpoint technique. The optimization is done again via Downhill Simplex. Both are described in detail in Chapter 3.

6.3 Experimental Results and Discussion

The experimentation for this module is set up in the same format as that of the eye module. There are two separate issues being investigated. The first aims at testing the detection capability of the mouth module given the proper context. The second intends to establish a quantitative measure by which a properly fitted mouth template can be distinguished from the incorrectly fitted ones.

Shape Term	Variable l	Spring Constants k_1	k_2	Ideal State l_1	l_2	Range from	to
X_{mouth}	$X(mouth)$	0.2	-	$xcenter$	-	$width * 0.3$	$width * 0.7$
Y_{mouth}	$Y(mouth)$	5.0	0.5	$ycenter - 2$	$ycenter + 4$	$ycenter - 2 - height * 0.3$	$ycenter + 4 + height * 0.3$
B_{mouth}	$B(mouth)$	4.0	1.0	$width * 0.3$	$width * 0.4$	$width * 0.2$	$width * 0.5$
$A1_{mouth}$	$A1(mouth)$	1.0	2.0	$B(mouth) * 0.25$	$B(mouth) * 0.3$	1	$width * 0.2$
$A2_{mouth}$	$A2(mouth)$	40.0	40.0	-2	2	$-width * 0.1$	$width * 0.1$
$A3_{mouth}$	$A3(mouth)$	1.0	2.0	$B(mouth) * 0.3$	$B(mouth) * 0.4$	1	$width * 0.25$
θ_{mouth}	$\theta(mouth)$	0.25	-	$\theta(line)$	-	$\theta(line) - 5$	$\theta(line) + 5$

where $N(mouth)$ is the N parameter of the mouth template;
$xcenter$ is the x coordinate of the center of the processing region;
$ycenter$ is the y coordinate of the center of the processing region;
$width$ is the width of the processing region;
$height$ is the height of the processing region;
$\theta(line)$ is the tilt angle of the reference line supplied by the Hough transform step.

Table 6.1: Mouth Shape Function Coefficients

Rating	Number	Percentage
Good	15	36
Adequate	19	45
Marginal	5	12
No Fit	3	7

Table 6.2: Mouth Module Results

In order to examine the first issue, we tested the mouth module with 42 images. This is a subset of the images we used in a similar experiment for the eye module. The reason is that our mouth template is designed to handle closed mouths and clean shaven individuals only, and therefore some of the images were excluded. The results are again evaluated subjectively using the same rating categories as that of the eye module. An example of each is shown in Figure 6.4. The results are summarized in Table 6.2.

Generally, the findings here are similar to that in the eye module. There were very few exactly fitted cases because actual mouths seldom conform to a perfect parabolic shape. However, a relatively **Good** to **Adequate** fit is certainly

(a) Good (b) Adequate

(c) Marginal (d) No Fit

Figure 6.4: Mouth Template Fit

achievable providing there is good image quality and resolution. Because only the first horizontal line is used from the list generated by the Hough transform step, there is no guarantee that it is in fact part of the mouth. Most **No Fit** cases were generated by those with the incorrectly selected mouth line. This finding again is the same as that in the eye module where the performance of the deformable template step suffers as a result of failing to select the proper iris pair in the earlier step. As we pointed out in section 5.3, this can be rectified once a robust qualitative measure is derived for distinguishing valid mouths from the invalid ones.

Although the percentages presented here are fairly similar to those from the eyes module, on the few occasions that the module failed, it did so because of a different reason - the lack of reliable edge information. Unlike the eyes, lips do not always appear as a clearly outlined object in an intensity image. This is particularly true for subjects with very thin and/or light colored lips, such as that in Figure 6.4 (c) and (d). Our model template is an edge based one, and in order for it to lock on successfully, it requires a certain amount of edge information present.

There are several ways of rectifying this problem. One, we could change our model from an edge based one to that of a different representation. The idea of intensity gradient direction map in [RS92] could be one such model. With this technique, segmentation is achieved by identifying regions of similar gradient directions, rather than magnitudes. An object will be defined in terms of distinct gradient direction patches. Hence, it is most suited for describing objects with little or no distinct edge. This is a considerable modification to the current model and until it is implemented and thoroughly tested, it will be difficult to predict whether any improvement can be achieved.

The second is to retain our current model, but improve the inputs. Currently, we derive our edge information from a single intensity image. Because of the lack of intensity contrast between the lips and skin tone, the edge information is often scant. The problem can become extremely ill-posed, especially if it is compounded with the presence of a beard or moustache whose strong edge presence is expected to distract the proper template placement. The easiest method to enrich an

intensity image is by adding color inputs. Konrad demonstrated a successful example of how color can be used to reduce the ill-posedness inherent in a two dimensional motion estimation problem [Kon92]. Since most lips appear as some shade of red, color could be extremely useful in extracting a reliable outline of the mouth. Nevertheless, more work has to be done in order to integrate color information into existing edge detection techniques.

Alternatively, we could add an extra color energy term in the template model. The idea is to maximize the *redness* within the area between the upper and lower parabolas. This is potentially useful not only for extracting red lips, but also for rectifying our second major source of problems: cases of thin lips. Currently, we impose only a low penalty cost for deformation [2] in the shape function in order to allow the lips to deform into a thin line in the absence of a better fit. Ideally, the tendency for the template to deform will be counteracted by existence of strong edge further out. However, this is not the case in practice, and the middle line often appears the strongest edge out of the three. Consequently if the deformation penalty is set too low, most lips will deform. Yet, if it is set too high, the template could not accommodate cases of thin lips. Therefore, the combination of insufficient edge information and large variation in lip thickness makes the deformation penalty one of the most extremely difficult measure to tune. The addition of a maximal *redness* requirement will serve as a counter measure to pull the lips back out if possible. Hence, it is not as important to tune the deformation penalty as precisely.

The reader should note that all of the above problems are not unique to our particular approach. In fact, we expect any edge based methods to experience similar difficulties. Therefore, all the suggested enhancements will be valid for any primarily edge based approach to mouth detection, such as the original deformable template method proposed by Yuille *et al.* .

In addition to being able to extract a mouth shape description from a valid region, we want to derive a quantitative means to distinguish **Good** to **Adequate** fits from those that are **Marginal** and **No Fit**. However, because of the

[2]The deformation penalty is controlled by the k_1 coefficient to the $A1_{mouth}$ $A2_{mouth}$ terms as listed in Table 6.1.

insufficient edge information issue pointed out in the previous paragraph, it will be difficult to expect the correctly fitted cases to exhibit a significantly lower template energy cost than the invalid ones. Nevertheless, the possibility should still be investigated. The experiment was set up so that in addition to the previous 42 mouth regions, 20 more were added. Some of the additional ones were taken from incorrectly identified contexts. The others were taken from correctly identified contexts of individuals with a beard or moustache. This increased the total number of **Marginal** to **No Fit** cases to approximately the same as that of the **Good** to **Adequate** cases. The distribution of the template energy cost for both is plotted and shown in Figure 6.5:

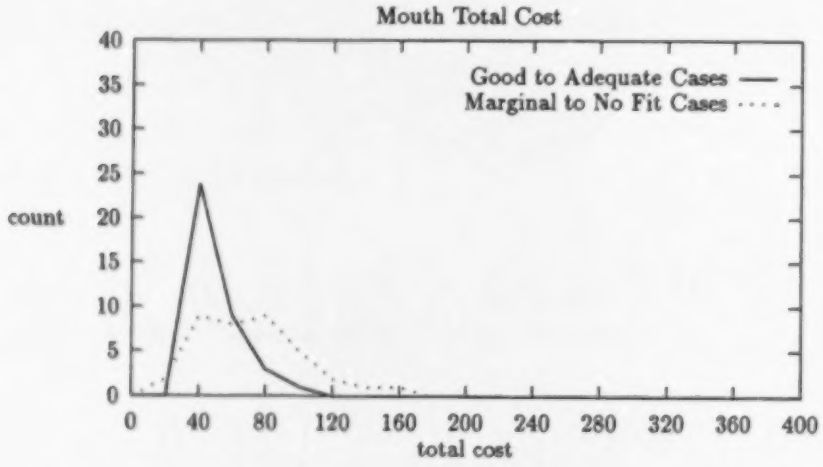


Figure 6.5: Mouth Template Energy Cost Distribution

One can see that there is no clear division between the two distribution. This again, reinforces our suspicion of inadequate contrast based only on intensity image. Though the correctly fitted cases have a much narrower distribution on the low end of the energy spectrum, it is not low enough to clearly distinguish itself from the invalid cases. In fact, the distribution of the incorrectly fitted cases is so wide that it overlaps the entire range of the valid cases. This module as is, is

clearly not robust enough. Perhaps the addition of color distinction will enhance the edge information and make this module more robust.

In addition to the detection performance, we measured the execution time performance of this module. The average run time was found to be 3.23 sec with a 0.570 sec standard deviation, a minimum of 2.270 sec and a maximum of 4.720 sec. This is again significantly faster than that reported by Yuille *et al.* . Some of it is of course due to the fact that we are using a slightly simpler template. However, we feel that much of it is due to the fact that care had been taken to code the geometric primitive generation and function optimization routines as efficiently as possible.

In summary, though enhancements must be added before this module can be used reliably as a confirmation module, it has demonstrated good potential in extracting mouth shape and location from well contrasted lips. As well, since the module can be executed in a very short time under a general purposed computing environment, it is potentially possible to be sped up even further with dedicated hardware.

Chapter 7

Conclusion

The problem of automatic facial feature extraction is a crucial, yet under-developed domain within the field of computerized face recognition. The difficulty is the lack of formalism in defining what a face is. Indeed, a face is a complex object comprised of components which although regular in general appearances, possess high degree of variation under detail examination.

The problems we are faced with in automatic facial feature extraction therefore exist on two levels. On the low level, we need to derive a model for individual features that is descriptive enough to embody the common shape, but yet flexible enough to handle some degree of variation. As well for practicality, the technique employing such a model must be executable within a reasonably short time. On the high level, we are concerned with the overall system integrity. Visual recognition is an ill-posed task. Since there is seldom a unique solution, we compromise by choosing the most plausible interpretation. Yet this choice could have been wrong. If this error is allowed to propagate and accumulate throughout a system, such as the case in a sequential design that is frequently employed by researchers in this area, the integrity of the overall system is highly questionable. Therefore, we wish to establish a system that will combine the results of individual feature extraction modules with minimum compounded error.

The research described in this thesis attempts to be the first step of constructing a system that would solve these two problems. The current system is comprised of three modules. These are the context module which generates hy-

potheses and provides the overall system control, and the eyes and mouth modules which confirm the existence and extract the precise shape and location of their respective features of interest.

The eyes and mouth modules employ a combined Hough transform and deformable template technique. Both of these techniques are high level modelling approaches which aim at identifying objects which can be decomposed into simple geometric elements. For the extraction purpose, the results were very favorable. The rate of extraction for *Adequate* or better is 84% for the eyes module, and 81% for the mouth module. For the context confirmation purpose, more work is clearly needed before the feature modules can be used reliably. The future work involved here would be primarily the investigation of additional energy terms for the deformable template model that will serve to better distinguish between valid and invalid cases.

The context module, on the other hand, relies on simple heuristics and rough segmentation by morphological filtering and blob coloring. It was capable of capturing the correct location in 96% of the test images with an average run time of less than 1 second, a very promising result. The test cases here are limited to only individuals without spectacles. Therefore, a potential improvement in this module would include an alternative approach to handle bespectacled cases. As well, we have to investigate how the context module should integrate the results from the various feature modules and the possibility of refining these results in an iterative process. The important contribution of this module is the fact that it provides a proper control structure under which feature extraction modules can operate economically and reliably. It is economical because rough location has already been established and futile search can be limited. It is reliable because interdependency among feature extraction modules are completely eliminated, and thereby preventing errors from being accumulated.

Indeed, this is far from being a complete system, however it does provide a good framework to further future research in automatic face features extraction. With the current system design, new modules for features, such as eyebrows, noses, or even moustaches, can easily be added. As well, existing feature modules can be completely redesigned if better techniques are derived. Both of these

can be accomplished without imposing drastic impact on the rest of the system. Hopefully, this will eventually lead to a completely automated facial recognition system.

Bibliography

[Ake84] J.R. Van Aken. An efficient ellipse drawing algorithm. *Computer Graphics and Art*, pages 24–35, September 1984.

[Bar89] Robert J. Baron. Strengths and weaknesses of computer recognition systems. In Andrew W. Young and Hadyn D. Ellis, editors, *Handbook of Research on Face Processing*, chapter 10, pages 507–18. Elsevier Science Publishers B.V., P.O. Box 1991, 1000 BZ Amsterdam, The Netherlands, 1989.

[BB82] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice-Hall Inc., 1982.

[Bie87] Irving Biederman. Recognition-by-components : A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[CEL87] I. Craw, H. Ellis, and J.R. Lishman. Automatic extraction of face-features. *Pattern Recognition Letters 5*, pages 183–87, 1987.

[CJCM86] S.R. Cannon, G.W. Jones, R. Campell, and N.W. Morgan. A computer vision system for identification of individuals. *IEEE IECON'86 Proceedings*, 1:347–51, 1986.

[Con86] Michael J. Conlin. High level vision using a rule-based language. *IEEE Proceedings*, pages 1153–57, 1986.

[CTB92] I. Craw, D. Tock, and A. Bennett. Finding face features. *ECCV92*, May 1992.

[FvFH90] James D. Foley, Andries vanDam, Steven K. Feiner, and John F. Hughes. *Computer Graphics Principles and Practice.* Addison Wesley Publishing Company, second edition, 1990.

[GHL71] A. Jay Goldstein, Leon D. Harmon, and Ann B. Lesk. Identification of human faces. *Proceedings of the IEEE*, 59(5):748–760, May 1971.

[GSS90] V. Govindaraju, S.N. Srihari, and D.B. Sher. A computational model for face location. In *Proceedings of the 3rd International Conference on Computer Vision* [GSSS89], pages 718–721.

[GSSS89] V. Govindaraju, D.B. Sher, R.K. Srihari, and S.N. Srihari. Locating human faces in newspaper photographs. *Proceeding of CVPR*, pages 549–554, 1989.

[Har76] L.D. Harmon. Automatic recognition of human face profiles. *Proceeding of 3rd Internation Joint Conference in Pattern Recognition*, pages 183–188, 1976.

[KLO81] T. Kohonen, P. Lehtio, and E. Oja. Storage and processing of information in distributed associative memory systems. In G.E. Hinton and J.A. Anderson, editors, *Parallel Models of Associative Memory.* Hillsdat : Lawrence Erlbaum Associates, 1981.

[Kon92] Janusz Konrad. Use of colour in gradient-based estimation of dense two-dimensional motion. *Proceedings of Vision Interface '92*, pages 189–194, May 1992.

[Lam87] L.C. Lambert. Evaluation and enhancement of the afit autonomous face recognition machine. Master's thesis, Air Force Institute of Technology, 1987.

[Mar87] Petros Maragos. Tutorial on advances in morphological image processing and analysis. *Optical Engineering*, 26(7):623–32, July 1987.

[Nix85] Mark Nixon. Eye spacing measurement for facial recognition. *SPIE Applications fo Digital Image Processing VIII*, 575:279–85, 1985.

[NR65] J.A. Nelder and R.Mead. *Computer Journal*, 7:308, 1965.

[Pit67] M.L.V. Pitteway. Algorithm for drawing ellipses or hyperbolae with a digital plotter. *Computer Journal*, pages 282–289, November 1967.

[RC92] Azriel Rosenfeld and Rama Chellappa. Computer vision : Attitudes, barriers, counseling. *Proceedings of Vision Interface '92*, pages 1–7, May 1992.

[RS92] Graham Robertson and Ken Sharman. Object location using proportions of the direction of intensity gradient (prodigy). *Proceedings of Vision Interface '92*, pages 189–194, May 1992.

[Sam91] A. Samal. Minimum resolution for human face detection and identification. *Proc. SPIE/SPSE Symp. Electronic Imaging*, January 1991.

[Ser82] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, 111 Fifth Avenue, New York, New York 10003, 1982.

[SK87] L. Sirobich and M. Kirby. Low-dimensional procedure for the characterization of human faces. In *Optical Society of America Journal* [TP89], pages 519–24.

[SNK72] Toshiyuki Sakai, Makoto Nagao, and Takeo Kanade. Computer analysis and classification of phototgraphs of human faces. *First USA-Japan Computer Conference Proceedings*, pages 55–62, October 1972.

[Sto86] J. Stonham. Practical face recognition and verification with wisard. In H. Ellis, M. Jeeves, F. Newcombe, and A. Young, editors, *Aspects of Face Processing.* Dordrecht: Martinus Nijhoff, 1986.

[SW91] M.A. Shackleton and W.J. Welsh. Classification of facial features for recognition. *Proceedigns of the IEEE Conference on Computer Vision and Pattern Recognition (CVIP-91)*, pages 573–79, 1991.

[TP89] Matthew Turk and Alex Pentland. Face processing: models for recognition. *SPIE Intelligent Robots and Computer Vision VIII: Algorithms and Techniques*, 1192:22–32, 1989.

[Vog89] Robert C. Vogt. *Automatic Generation of Morphological Set Recognition Algorithms.* Springer-Verlag New York Inc., 1989.

[Wel88] Gary L. Wells. *Eyewitness Identification.* The Carswell Co. Ltd., 2330 Midland Avenue, Agincourt, Ontario, Canada, M1S 1P7, 1988. Psychology account of techniques used in eyewitness identification.

[WLT89] K.H. Wong, H.M. Law, and P.W.M. Tsang. A system for recognising human faces. *International Conference on Acoustics, Speech and Signal Processing Proceedings*, III:1638–42, 1989.

[YCH88] Alan Yuille, David Cohen, and Peter Hallinan. Facial feature extraction by deformable templates. Technical Report 88-2, Harvard Robotics Laboratory, 1988.

Appendix A

Modified Midpoint Circle Algorithm

Since the Midpoint circle algorithm is integer-based, it naturally expects all its parameters (ie. center coordinates, and radius) to be integer as well. This means that only a full circle with integer coordinates and even diameter can be generated. Yet, a circle in an image may have either an odd or even diameter. With an odd diameter circle, the center will be located half way in between two pixels rather than on the pixel. Therefore, though the circle boundary points (ie. the edge pixel) may have integer coordinates, the corresponding parameter cell could have real number coordinates center. We could of course use a less efficient floating-point version of the Midpoint circle algorithm where pixel coordinates must be generated using floating point arithmetic even though they will eventually be rounded off as integers.

Therefore, instead of one general circle generating algorithm, we decide to classify circles into even or odd and have two separate routines handling them. We recall that the octants can be generated from symmetry. This is to say that for every point within an octant there exist 7 corresponding points, one in each octant as shown in Figure A.1.

Although the overall symmetry is slightly different between the two, the locations of points within each octant are very similar as shown in Figure A.2. In fact, the octants of an odd diameter circle can be generated from those of a slightly

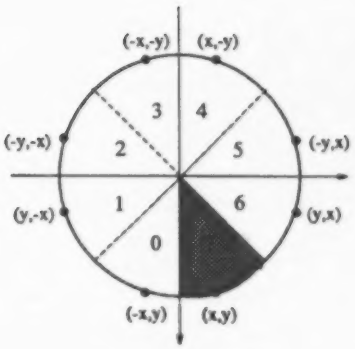


Figure A.1: Eight Symmetrical Points on a Circle

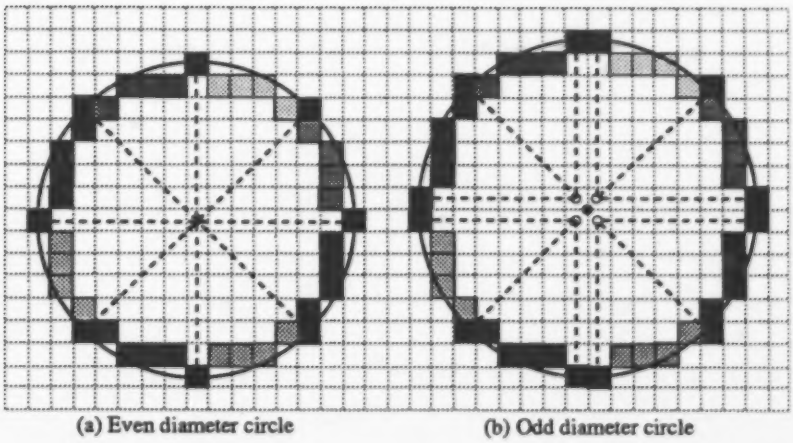


Figure A.2: Even and Odd Circles

Octant	Even-Diameter Circle		Odd-Diameter Circle	
	Δx	Δy	Δx	Δy
0	$-x$	y	y	$-x-1$
1	$-y$	x	x	$-y-1$
2	$-y$	$-x$	$-x-1$	$-y-1$
3	$-x$	$-y$	$-y-1$	$-x-1$
4	x	$-y$	$-y-1$	x
5	y	$-x$	$-x-1$	y
6	y	x	x	y
7	x	y	y	x

Notes: Δx and Δy are the amount to add to the truncated x and y coordinates of the circle center.

Table A.1: Generation of Symmetrical Points on a Circle

smaller even diameter circle just by shiftying each octant out by half a pixel from the center. Yet, since the odd diameter circle is larger, the total number of points lie on its circumference should be more than that of the smaller even diameter circle. This difference can be accounted for by adding additional symmetrical points for pixels lying at exactly 0, $\frac{\pi}{2}$, π, and $\frac{3\pi}{2}$ radians.

Hence, the overall strategies for both are effectively the same as the original Midpoint circle algorithm. The only difference lies in how symmetric points are calculated. Table A.1 summarizes the rules used to generate these symmetrical points for the two.

Appendix B

Circle Arc Generation

Often, one would want to generate only a partial rather than a complete circle, such as in the case of circle Hough transform with gradient information. Here, the midpoint circle algorithm can still be used, except a decision has to be made whether a pixel falls between the starting and ending points. Considering the potentially large number of circles one might have to generate, the time required may amount to a considerable sum.

However, with careful examination of the relations between a circle and its arc, we observe the followings:

1. An octant can be classified into four different types: starting, ending, interior, and exterior.
2. Interior octant is completely inside, therefore any point falling within it will be included without exception.
3. Exterior octant is completely outside, therefore any point falling outside it will be excluded without exception.
4. Starting octant is only partial covered, a comparison with starting point is necessary to determine whether it should be included.
5. Ending octant is only partial covered, a comparison with ending point is necessary to determine whether it should be included.

Therefore, an octant table can be used to classify each octant into one of the three categories: the beginning, ending and interior octants. Since we are using symmetry to generate points, we already know which octant they originate from. From the octant table we can then decide whether further comparison is needed. In effect, the modified algorithm drastically reduces the amount of comparisons, yet preserves the elegance and efficiency of the original midpoint circle generation technique.

Appendix C

Parabola Generation

The midpoint conic generation routine in [FvFH90] expresses a general conic in the form of:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \tag{C.1}$$

Equation C.1 assumes that the coefficients have been adjusted so that the starting point is at the origin. This is needed for handling partial conics (ie. an elliptical arc, or finite parabola). In order to use the Midpoint conic routine for generating parabolas, we must derive an expression for each coefficient in equation C.1 in terms of the parabola parameters as shown in Figure C.1:

The analytic equation for this parabola can be expressed as:

$$y' = \frac{-a}{b^2}x'^2 + a \tag{C.2}$$

with $E_0 = (-b, 0)$ and $E_1 = (b, 0)$.

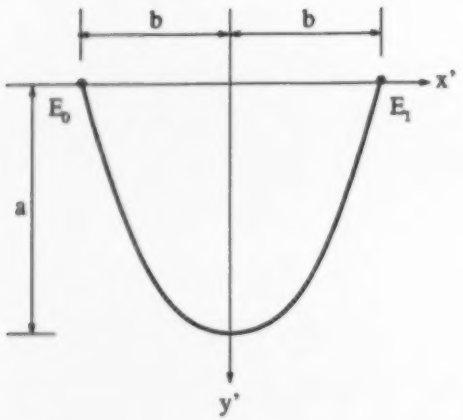
If the parabola is now rotated by an angle of θ as shown in Figure C.2: The relations between the parabola coordinates (x', y') and the image plane coordinates (x, y) are as follows:

$$x' = \cos\theta x - \sin\theta y \tag{C.3}$$

$$y' = \sin\theta x + \cos\theta y \tag{C.4}$$

$$x = \cos\theta x' + \sin\theta y' \tag{C.5}$$

$$y = \cos\theta y' - \sin\theta x' \tag{C.6}$$

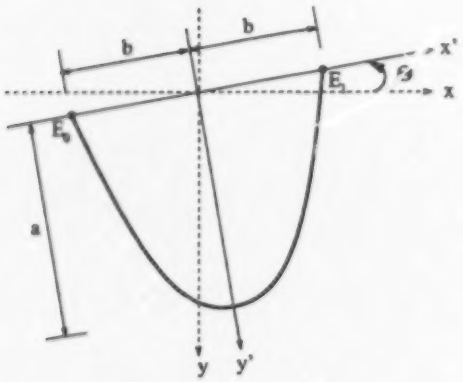


Legend :

a - the height of the parabola

b - the half-width of the parabola

Figure C.1: Upright Parabola



Legend :

a - the height of the parabola

b - the half-width of the parabola

Θ- the tilt angle

Figure C.2: Rotated Parabola

Therefore the rotated endpoints are now:

$$E_0 = (-b\cos\theta, b\sin\theta)$$
$$E_1 = (b\cos\theta, -b\sin\theta)$$

Substituting equations C.3 and C.4 into equation C.2, we obtain:

$$\begin{aligned}
\sin\theta x + \cos\theta y &= \tfrac{-a}{b^2}(\cos\theta x - \sin\theta y)^2 + a \\
b^2\sin\theta x + b^2\cos\theta y &= -a(\cos^2\theta x^2 - 2\cos\theta\sin\theta xy + \sin^2\theta y^2) + ab^2 \qquad \text{(C.7)} \\
a\cos^2\theta x^2 - 2a\cos\theta\sin\theta xy + a\sin^2\theta y^2 &+ b^2\sin\theta x + b^2\cos\theta y - ab^2 = 0
\end{aligned}$$

In terms of C.1, we have:

$$A = a\cos^2\theta \qquad \text{(C.8)}$$
$$B = -2a\cos\theta\sin\theta \qquad \text{(C.9)}$$
$$C = a\sin^2\theta \qquad \text{(C.10)}$$
$$D = b^2\sin\theta \qquad \text{(C.11)}$$
$$E = b^2\cos\theta \qquad \text{(C.12)}$$
$$F = -ab^2 \qquad \text{(C.13)}$$

Now, translating E_1 to the origin, we obtain the following transformation of coordinates:

$$x \Rightarrow x + b\cos\theta \qquad \text{(C.14)}$$
$$y \Rightarrow y - b\sin\theta \qquad \text{(C.15)}$$

Substituting equations C.14 and C.15 into equation C.1, we have:

$$\begin{aligned}
&A(x + b\cos\theta)^2 + B(x + b\cos\theta)(y - b\sin\theta) \\
&\quad +C(y - b\sin\theta)^2 + D(x + b\cos\theta) + E(y - b\sin\theta) + F = 0 \\
&A(x^2 + 2b\cos\theta x + b^2\cos^2\theta) \\
&\quad +B(xy - b\sin\theta x + b\cos\theta y - b^2\sin\theta\cos\theta) \\
&\quad +C(y^2 - 2b\sin\theta y + b^2\sin^2\theta) + D(x + b\cos\theta) \qquad \text{(C.16)} \\
&\quad +E(y - b\sin\theta) + F = 0 \\
&Ax^2 + Bxy + Cy^2 + (2b\cos\theta A - b\sin\theta B + D)x + (b\cos\theta B \\
&\quad -2b\sin\theta C + E)y + (b^2\cos^2\theta A - b^2\cos\theta\sin\theta B \\
&\quad +b^2\sin^2\theta C + b\cos\theta D - b\sin\theta E + F) = 0
\end{aligned}$$

Therefore, the new coefficients are:

$$A' = A = a\cos^2\theta \tag{C.17}$$

$$B' = B = -2a\cos\theta\sin\theta \tag{C.18}$$

$$C' = C = a\sin^2\theta \tag{C.19}$$

$$\begin{aligned} D' &= 2b\cos\theta A - b\sin\theta B + D \\ &= 2b\cos\theta(a\cos^2\theta) - b\sin\theta(-2a\cos\theta\sin\theta) + (b^2\sin\theta) \\ &= 2ab\cos^3\theta + 2ab\cos\theta\sin^2\theta + b^2\sin\theta \\ &= 2ab\cos\theta(\cos^2\theta + \sin^2\theta) + b^2\sin\theta \\ &= 2ab\cos\theta + b^2\sin\theta \end{aligned} \tag{C.20}$$

$$\begin{aligned} E' &= b\cos\theta B - 2b\sin\theta C + E \\ &= b\cos\theta(-2a\cos\theta\sin\theta) - 2b\sin\theta(a\sin^2\theta) + (b^2\cos\theta) \\ &= -2ab\cos^2\theta\sin\theta - 2ab\sin^3\theta + b^2\cos\theta \\ &= -2ab\sin\theta(\cos^2\theta + \sin^2\theta) + b^2\cos\theta \\ &= -2ab\sin\theta + b^2\cos\theta \end{aligned} \tag{C.21}$$

$$\begin{aligned} F' &= b^2\cos^2\theta A - b^2\cos\theta\sin\theta B + b^2\sin^2\theta C + b\cos\theta D - b\sin\theta E + F \\ &= b^2\cos^2\theta(a\cos^2\theta) - b^2\cos\theta\sin\theta(-2a\cos\theta\sin\theta) + b^2\sin^2\theta(a\sin^2\theta) \\ &\quad + b\cos\theta(b^2\sin\theta) - b\sin\theta(b^2\cos\theta) + (-ab^2) \\ &= ab^2\cos^4\theta + 2ab^2\cos^2\theta\sin^2\theta + ab^2\sin^4\theta \\ &\quad + b^3\cos\theta\sin\theta - b^3\cos\theta\sin\theta - ab^2 \\ &= ab^2(\cos^2\theta + \sin^2\theta)^2 - ab^2 \\ &= ab^2 - ab^2 \\ &= 0 \end{aligned} \tag{C.22}$$

Given the height a, half-width b, and tilt angle θ of a parabola, equations C.17 - C.22 can be used to compute the necessary coefficients for the midpoint conic generation algorithm. Note that, though the center of the parabola is assumed to lie on the origin here, a simple offset can be added to all points to accommodate the general cases.

Appendix D

Image Processing Library

During the course of this study, many routines were developed for various tasks, such as image processing, graphics, data structure management etc.. Most were designed with general purpose in mind and form a collection of useful routines callable from any "C" programs. Here, we shall give an overview of what functionalities these routines provide.

Image Structure Management

InitByteImage initializes a *ByteImage* structure to the specified dimension.

FreeByteImage frees the memory allocated for the specified *ByteImage*.

InitRealImage initializes a *RealImage* structure to the specified dimension.

FreeRealImage frees the memory allocated for the specified *RealImage*.

CopyRealImage copies a block of pixels values from one *RealImage* to another.

InitCompImage initializes a *CompImage* (ie. frequency domain image) structure to the specified dimension.

FreeCompImage frees the memory allocated for the specified *CompImage*.

CopyCompImage copies a block of complex values from one *CompImage* to another.

Byte2Real converts from a *ByteImage* to a *RealImage*.

Real2Comp converts from a *RealImage* to a *CompImage* by filling the imaginary components with 0's.

Comp2Real converts from a *CompImage* to a *RealImage* by truncating the imaginary components.

Image structure and raster file I/O

Real2RaswCTBL outputs a *RealImage* into a sun raster file with the specified color table.

Ras2Real inputs a *RealImage* from a sun raster file assuming 256 grey scales.

Real2Ras writes a 256 grey scale *RealImage* as a sun raster file.

Mult2Ras overlays a grey scale image with a number of binary mask images with the specified colors and outputs it as a sun raster.

Spatial Domain Image Manipulation

ScaleImage modifies the pixels value of a *RealImage* so that it is within the specified range of grey scales.

ScaleByteImage modifies the pixels value of a *ByteImage* so that it is within the specified range of grey scales.

ZeroCrossing locates the zero crossing points in a Laplacian filtered image.

SobelEdges calculates the edge image using Sobel filtering.

GaussEdgeLoc calculates the edge image using 2nd Derivative Gaussian filtering.

LapEdgeLoc determines the location of edges by Laplacian filtering.

Sample converts the input image into a smaller version by sampling every N pixels column and row wise as specified.

Threshold performs range thresholding for a *RealImage*.

SubRealImages subtracts two *RealImage*.

MaskRealImage masks a *RealImage* with a binary *RealImage*

SpatialFilter performs spatial filtering with the supplied filter function on a *RealImage*.

Dilate a morphological dilation filter function (to be used with *SpatialFilter*).

Erode a morphological erosion filter function (to be used with *SpatialFilter*).

Med a median filter function (to be used with *SpatialFilter*).

Convolve a spatial convolution filter function (to be used with *SpatialFilter*).

Frequency Domain Image Manipulation

ComAbs returns the absolute value of a complex number.

ComMult multiplies two complex numbers.

ComDiv divides one complex number by another.

ComAdd adds two complex numbers.

ComSub subtracts two complex numbers.

fft2d performs a 2-D Fast Fourier Transform on the input *RealImage* and returns its *CompImage* counterpart.

ZeroPhaseFilter performs frequency filtering with the supplied zero phase filter function on a *CompImage*.

BLPF a Butterworth low pass filter function (to be used with *ZeroPhaseFilter*).

ILPF an ideal low pass filter function (to be used with *ZeroPhaseFilter*).

GaussLPF a Gaussian low pass filter function (to be used with *ZeroPhaseFilter*).

Gauss2nd a second derivative Gaussian filter function (to be used with *ZeroPhaseFilter*).

PhSh90Filter performs frequency filtering with the supplied 90° phase shifted filter function on a *CompImage*.

Gauss1st a first derivative Gaussian filter function (to be used with *PhSh90Filter*).

Gauss1x a first derivative (with respect to x) Gaussian filter function (to be USED with *PhSh90Filter*).

Gauss1y a first derivative (with respect to y) Gaussian filter function (to be used with *PhSh90Filter*).

Region Segmentation

BlobInit initializes storages for region management.

BlobNew creates an id for a new region.

BlobAdd adds a pixel to the specified region.

BlobGetID returns the id of the region to which the specified pixel belongs.

BlobConnect connects two regions and marks their id's as equivalent.

BlobClose compresses all connected region and returns a condensed list of regions and the corresponding segmented image colored by regions.

Blob2Ras outputs the segmented image to a sun raster file.

GetBlobs performs blob coloring with the given image and returns the segmented image and a condensed list of regions.

Geometric Primitive Generation

ScanArc generates a circular arc.

ScanCircle generates a complete circle.

ScanConic generates a conic.

ScanEllipse generates an ellipse.

ScanParabola generates a symmetric parabola.

ScanLine generates a line.

Drawing Routines

PutCross draws a cross.

PutBox draws a box.

DrawArc draws a circular arc.

DrawCircle draws a complete circle.

DrawEllipse draws an ellipse.

DrawParabola draws a symmetric parabola.

DrawLine draws a line.

Miscellaneous Utilities

StartClock starts timing for the current nesting level and increments the level hereafter.

StopClock stops the current nesting level timing and decrements the level hereafter.

PrintClock reports the timing so far for the current nesting level.

InitStat initializes for beginning statistic bookkeeping.

AddStatItem adds item to the statistics.

EndStat returns the mean, variance, maximum and minimum among all items added.

Get2DArray allocates the storage for a 2-D array.

Free2DArray frees the allocated memory for a 2-D array.

Get3DArray allocates the storage for a 3-D array.

Free3DArray frees the allocated memory for a 3-D array.

SetPercentile calculates the percentile for the given image region.

Per2Edge returns the corresponding pixel magnitude for the given percentile (called after *SetPercentile* has been executed).

Edge2Per returns the corresponding percentile for the given pixel magnitude (called after *SetPercentile* has been executed).

PercentRealImage converts all pixels in an image to their respective percentiles.

SimplexMinimize performs downhill simplex optimization.

HoughCircles performs circle Hough transform.

HoughLines performs line Hough transform.